Ведь это так просто!

Android™ Разработка приложений для ЧАЙНИКОВ®

Научитесь:

- создавать приложения для современных смартфонов Droid X, Galaxy S и MyTouch
- загружать SDK и конфигурировать среду Eclipse
- программировать приложения для Android
- публиковать свои приложения на сайте Android Market

Донн Фелкер

Android™ Разработка приложений для ЧАЙНИКОВ®

Android™ Application Development FOR DUMMIES®

by Donn Felker
with Joshua Dobbs

Wiley Publishing, Inc.

Android™ Разработка приложений для ЧАЙНИКОВ®

Донн Фелкер
при участии Джошуа Доббса

ДИАЛЕКТИКА
Москва ◆ Санкт-Петербург ◆ Киев
2012

ББК 32.973.26-018.2.75
Ф38
УДК 681.3.07

Компьютерное издательство “Диалектика”
Главный редактор *С.Н. Тригуб*
Зав. редакцией *В.Р. Гинзбург*
Перевод с английского и редакция канд. техн. наук *А.Г. Сысонюка*

По общим вопросам обращайтесь в издательство “Диалектика” по адресу:
info@dialektika.com, http://www.dialektika.com

Фелкер, Донн.
Ф38 Android: разработка приложений для чайников. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2012. — 336 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1748-5 (рус.)

ББК 32.973.26-018.2.75

Научно-популярное издание

Донн Фелкер

Android: разработка приложений для чайников

В издании использованы карикатуры американского художника Рича Теннанта

Литературный редактор *Е.П. Перестюк*
Верстка *Л.В. Чернокозинская*
Художественный редактор *В.Г. Павлютин*
Корректор *Л.А. Гордиенко*

Подписано в печать 08.11.2011. Формат 70x100/16
Гарнитура Times. Печать офсетная
Усл. печ. л. 27,09. Уч.-изд. л. 18,73.
Тираж 1500 экз. Заказ № 0000

Отпечатано с готовых диапозитивов
в ГУП “Типография «Наука»”
199034, Санкт-Петербург, 9-я линия В. О., 12.

ООО “И. Д. Вильямс”, 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1748-5 (рус.)

ISBN 978-0-470-77018-4 (англ.)

Оглавление

Содержание

Об авторе

Донн Фелкер — специалист в области разработки программного обеспечения для Интернета и мобильных устройств, независимый консультант с более чем десятилетним опытом. Основатель софтверной компании Agilevent, обладатель сертификатов Microsoft ASP Insider, MCTS (.NET Framework 2.0/3.5 Web Applications) и ScrumMaster. Автор учебного видеокурса TekPub.com "Introduction to Android". Блог Донна находится по адресу `blog.donnfelker.com`, а микроблог на Твиттере имеет адрес `@donnfelker`.

О соавторе

Джошуа Доббс — главный разработчик веб-приложений в компании, выпускающей электронные устройства и расположенной в Южной Калифорнии. Имеет более чем десятилетний опыт разработки настольных и веб-приложений. С операционной системой Android начал работать, как только она появилась на рынке программного обеспечения. Разработанные им приложения для Android загружены пользователями более 6 миллионов раз. Был выбран компанией Google в качестве главного разработчика приложений Android для проекта Device Seeding Program. Веб-сайт Джошуа находится по адресу `www.joshdobbs.com`.

Введение

Перед вами первая книга серии ...*для чайников*, посвященная разработке приложений для Android! Когда мне предложили ее написать, я пришел в восторг от возможности переложить на бумагу огромное количество знаний об Android, которые я приобрел за последние несколько лет. Но одновременно я задумался: а можно ли сделать столь сложную тему доступной для "чайников"? Оказывается, можно! Хорошо подобранные примеры и тщательно описанные инструкции позволят любому человеку, умеющему водить мышкой по коврику, создать свою первую программу для Android. Надеюсь, вы получите от создания своей первой программы для Android не меньшее удовольствие, чем то, которое я получал в процессе написания книги.

Когда компания Google приобрела операционную систему Android в 2005 году (вначале она была разработана небольшой компанией), я, честно говоря, не обратил на нее внимания. Я слышал, что Google планирует выйти на рынок мобильных устройств, но, как и во всем, что касается индустрии программного обеспечения, я не верил этому, пока не увидел первый результат — мобильный телефон G1 на основе операционной системы Android. Как только я услышал об этом, я "приклеился" к компьютеру, изучая новый продукт и собирая о нем всю информацию, которую только мог раздобыть.

Я начал разрабатывать свои первые приложения для Android приблизительно через неделю, после того как жена купила устройство G1. Это было первое устройство Android, поступившее в продажу. В то время оно не предоставляло такого богатого набора средств, как iPhone, но тем не менее я отчаянно верил в успех платформы Android. Когда была выпущена версия Donut (Android 1.6), стало очевидно, что Google продвигает продукт и не жалеет финансовых ресурсов. Сразу после версии 1.6 начались разговоры о версии 2.0, а это говорит о том, что ее ждали с нетерпением.

На момент написания данной книги наиболее популярна версия 2.2, причем версия 3.0 уже на подходе[1]. Платформе уже несколько лет, но усовершенствования продолжаются. Несомненно, это наиболее захватывающий период для разработки приложений Android. Надеюсь, при чтении книги вам передастся воодушевление всех людей, причастных к данному процессу.

Предназначение книги

Книга написана для начинающих пользователей. Чтобы ее изучать, не обязательно иметь опыт разработки приложений Android или знать что-либо об Android. Предполагается, что читатель начинает с чистого листа. На данный момент существует много инструментов и технологий разработки приложений Android. Большинство профессиональных программистов используют в своей повседневной работе какую-либо одну технологию и часто ничего не знают об остальных. Мы сделаем так же: обсудим одну из наиболее популярных технологий на основе инструмента Eclipse и эмулятора мобильных устройств. Предполагается, что вы знаете элементы языка программирования Java. Впрочем, ни опыт работы, ни глубокое понимание Java для чтения книги не обязательны. Вам достаточно быть знакомыми с базовыми конструкциями языка.

---

[1] На момент издания русскоязычного перевода основной версией, используемой на смартфонах, была 2.3, а на плантшетных ПК — 3.0. — *Прим. ред.*

Все остальное, что понадобится для выполнения упражнений, вы узнаете из книги. Кроме того, при разработке приложений Android используется язык XML, однако и о нем вам нужно иметь лишь элементарное представление, потому что Eclipse автоматически генерирует документы XML, и вам остается только подставлять нужные значения атрибутов. Например, Eclipse не знает, какого цвета кнопка вам нужна, и по умолчанию делает ее серой. Замените слово `gray` (серый) значением `red` (красный) или `green` (зеленый), и можно считать, что вы умеете программировать для Android.

Платформа Android позволяет разрабатывать приложения для устройств самых разных типов, включая мобильные телефоны, электронные книги, плееры, нетбуки, устройства GPS (Global Positioning System — система глобального позиционирования) и др. В недалеком будущем этот список пополнится телевизорами — невероятно, правда?! Компания Google уже объявила о планах включить телевидение в набор своих услуг на платформе Android.

Конечно, прочитав эту книгу, вы не станете крупным специалистом в области разработки приложений для Android, но перед вами откроются широкие возможности. Запрограммировав свое первое приложение, вы создадите прочный фундамент для совершенствования мастерства и приобретения новых знаний. Данная книга представляет собой "концентрат" сотен или даже тысяч страниц технической документации по Android. Чтобы написать ее, я прочитал десятки учебников и извлек из них наиболее полезные методики, советы и трюки. Не воспринимайте книгу как сборник рецептов. Она предназначена для того, чтобы помочь вам собрать многочисленные фрагменты инфраструктуры Android в работоспособное интерактивное приложение.

Соглашения, используемые в книге

По мере работы над книгой вы будете использовать классы инфраструктуры Android и создавать классы Java и файлы XML.

Примеры кодов набраны моноширинным шрифтом, чтобы их можно было легко отличить от текста книги. Ниже приведен пример кода.

```
public class MainActivity
```

Высокоуровневый язык программирования Java чувствителен к регистру, поэтому будьте внимательны и вводите текст в редакторе точно так же, как он приведен в книге. Кроме того, в книге используются стандартные соглашения Java, поэтому вам будет легко перейти от примеров книги к примерам кода, представленным в Android SDK (Software Development Kit — набор инструментов для разработки приложений). В частности, все имена классов приводятся в формате `PascalCase`. Все переменные уровня класса начинаются с префикса `m`.

Моноширинным шрифтом набраны также все URL-адреса, например:

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

Если вы не уверены в правильности какого-либо фрагмента кода, можете загрузить исходные коды по адресу `http://github.com/donnfelker`. Время от времени я обновляю исходные коды, но после каждого обновления они работоспособны и полностью согласованы с текстом книги. Их можно найти также на американском сайте книги по адресу `http://www.dummies.com/go/androidappdevfd`.

Основные предположения

Для программирования приложений Android вам необходим компьютер, на котором установлена любая из приведенных ниже операционных систем:

- ✓ Windows XP (32-разрядная), Vista (32- или 64-разрядная) или Windows 7 (32- или 64-разрядная);
- ✓ Mac OS X (Intel) 10.5.8 (только x86);
- ✓ Linux (i386).

В книге предполагается, что на вашем компьютере установлена Windows, но даже если установлена другая операционная система, для вас почти ничего не изменится: другими будут только окна конфигурирования среды и каталоги установки инструментов.

Вам нужно также загрузить бесплатные пакеты Android SDK и JDK (если их еще нет на вашем компьютере). Подробные инструкции по инсталляции этих инструментов приведены в главе 2.

Приложения Android разрабатываются на языке Java, поэтому вам нужно быть знакомым с базовыми элементами синтаксиса Java. Кроме того, в ресурсах Android часто используются файлы XML, поэтому вам нужно знать также базовые элементы синтаксиса XML. Впрочем, вам не нужно быть специалистом по XML и Java. Если вы знаете любой другой язык программирования (например, C#), этого вполне достаточно для усвоения материала книги.

Не требуется также иметь на руках физическое устройство Android, потому что создать и отладить приложение Android можно с помощью эмулятора устройств. Тем не менее, если у вас есть возможность, приобретите и используйте реальное физическое устройство, чтобы почувствовать себя на месте пользователя.

Структура книги

Книга состоит из четырех частей.

Часть I. Начальные сведения об Android

В части I приведено описание инструментов и инфраструктур, используемых для разработки приложений Android, и даны инструкции по их загрузке из Интернета и установке на настольный компьютер. Кроме того, в этой части представлены различные компоненты SDK и рассмотрены способы их использования на платформе Android.

Часть II. Создание и публикация приложения Android

В этой части вы создадите свое первое приложение Android и установите его на реальное или виртуальное устройство. Вы увидите созданное вами приложение на экране устройства Android и сможете поэкспериментировать с ним в интерактивном режиме. Кроме того, в этой части демонстрируется публикация созданного вами приложения в Android Market (конечно, пока что это не означает, что вы заработаете реальные деньги, продав свое приложение, но со временем возможно и такое).

Часть III. Создание мощных приложений

В части III вы повысите свою квалификацию, создав "напоминалку" — приложение, напоминающее в определенный момент времени о задачах и работах, которые необходимо сделать (на случай, если вы о них забыли). В данной части рассматривается реализация базы данных SQLite в многоэкранном приложении. Кроме того, вы узнаете об использовании строки состояния Android для отображения извещений, помогающих повысить удобство использования приложения.

Часть IV. Великолепные десятки

В этой части приведены полезные советы и трюки, которые я нашел методом проб и ошибок за годы работы с Android. Кроме того, в этой части приведен обзор шаблонов, на основе которых вы сможете создавать собственные, весьма мощные приложения. Приведены также описания полезных библиотек Android, которые существенно облегчат вашу карьеру профессионального разработчика.

Пиктограммы, используемые в книге

Эта пиктограмма обозначает полезные советы, которые желательно не пропустить.

Этой пиктограммой обозначены сведения, которые нужно помнить, чтобы понимать излагаемый далее материал.

Абзацы, отмеченные этой пиктограммой, содержат интересные сведения, которые, однако, не обязательны для понимания рассматриваемых технологий разработки приложений Android. Если вы не любопытны или у вас мало времени, можете пропустить эти абзацы.

Этой пиктограммой отмечены потенциальные проблемы, с которыми можно столкнуться в процессе разработки приложения. Прочитайте и запомните эти абзацы, иначе, когда вы столкнетесь с указанной проблемой, на ее устранение уйдет намного больше времени.

Что дальше

Итак, наступило время изучить платформу Android. Если вы нервничаете из-за ее сложности, то, уверяю вас, напрасно. У вас все получится. Для этого разделите работу над каждой задачей на два этапа. На первом точно выполняйте мои инструкции, чтобы заставить приложение заработать, а на втором этапе, когда оно заработало, смело экспериментируйте с ним, проверяя любые, даже самые сумасшедшие, идеи. Зайдя в тупик, вы в любой момент можете вернуться к работающему приложению и снова начать эксперименты.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail:	info@dialektika.com
WWW:	http://www.dialektika.com

Наши почтовые адреса:

в России:	127055, Москва, ул. Лесная 45, стр. 1
в Украине:	03150, Киев, а/я 152

Часть I

Начальные сведения об Android

The 5th Wave

Рич Теннант

"Знаешь, идея полностью беспроводного будущего меня жутко пугает".

В этой части...

В части I дается общее описание платформы Android и кратко рассмотрены ее основные характеристики. Кроме того, описаны инструменты Android SDK и способы их применения для создания приложений Android, а также даны подробные инструкции по установке и конфигурированию Android SDK.

Глава 1

Краткий обзор платформы Android

В этой главе...

- Зачем разрабатывать приложения для Android
- Основы программирования для Android
- Инструменты для работы с оборудованием
- Программные инструменты и ресурсы

Когда в 2005 году компания Google приобрела проект Android, все были очень удивлены и не верили в успех этого предприятия. Зачем успешной компании связываться с непрофильным сегментом рынка, перспективы на котором весьма туманные? Тем не менее время подтвердило правильность стратегии Google, которая доказала, что мобильная операционная система может разрабатываться и поддерживаться на открытой платформе. В настоящее время Google продолжает вкладывать деньги и ресурсы в проект Android, оказавшийся чрезвычайно успешным. В июле 2010 года ежедневно активировалось 160 тысяч мобильных устройств Android, что очень неплохо, учитывая, что первые устройства Android появились только в октябре 2008 года. Менее чем за два года Google радикально изменила ситуацию на рынке мобильных устройств!

Благодаря открытости платформы разработчикам еще никогда не было так легко зарабатывать деньги самостоятельно. Пользователи Android ничего не знают о вас, но они знают компанию Google и доверяют ей. Любой желающий может опубликовать свое приложение на сайте Android Market, контролируемом компанией Google, которая гарантирует качество приложений. Естественно, пользователи охотно покупают приложения на этом сайте.

Зачем разрабатывать приложения для Android

А почему бы и нет? Хотите, чтобы ваше приложение стало доступным для миллионов пользователей во всем мире? Хотите опубликовать свое приложение сразу после того, как оно будет отлажено и протестировано? Хотите заработать деньги интеллектуальным трудом? Хотите работать на открытой платформе? Если ответ хотя бы на один вопрос положительный, значит, вам имеет смысл заняться разработкой приложений Android. Если же вы все еще не решились, продолжайте читать книгу, и вам будет легче принять решение, вооружившись знаниями.

Сегмент рынка

Как разработчик, вы получаете благоприятную возможность создавать приложения для совершенно нового рынка, ежедневно растущего, как на дрожжах. Согласно прогнозам, ближайшие несколько лет Android будет обгонять другие платформы на рынке

программного обеспечения для мобильных устройств. При таком большом количестве пользователей еще никогда не было так легко писать приложения, которые будут загружаться реальными людьми. Служба Android Market передаст ваше приложение прямо в руки пользователей. Пользователям не нужно блуждать по Интернету в поисках нужного приложения, которое можно загрузить и установить. Служба Android Market предварительно установлена на всех устройствах Android (за несколькими исключениями, которые мы обсудим позже), поэтому пользователи обычно ищут нужные им приложения не в Интернете, а в Android Market. На сайте Android Market видно, что количество загрузок почти каждого приложения увеличивается ежедневно.

Время продавать

В составе платформы Android поставляется огромное количество библиотек API (Application Programming Interface — интерфейс программирования приложений). С их помощью легко разработать полнофункциональное приложение за относительно короткое время. Зарегистрировавшись в Android Market, выгрузите и опубликуйте ваше приложение. В отличие от других торговых сайтов, в Android Market нет стадии утверждения продукта. Вам достаточно написать приложение и опубликовать его.

Технически любой человек может опубликовать что угодно, однако Google настоятельно рекомендует придерживаться установленных правил и публиковать приложения в принятом формате. Не забывайте, что пользователи Android находятся во всех уголках земного шара и охватывают все возрастные, профессиональные, этнические и другие категории.

Открытая платформа

Операционная система Android является открытой платформой. Это означает, что она не привязана к одному производителю оборудования или провайдеру. Открытость платформы позволяет Android быстро завоевывать рынок. Все производители оборудования и провайдеры могут изготавливать и продавать устройства Android. Исходный код Android находится по адресу `http://source.android.com` и доступен для просмотра и модификации каждым желающим. Ничто не запрещает вам изучить исходный код, чтобы проанализировать, как решается интересующая вас задача. Открытость исходного кода позволяет производителям мобильных телефонов создавать для своих устройств удобные пользовательские интерфейсы и встраивать в них любые средства. Это ставит всех разработчиков в одинаковые конкурентные условия.

Происхождение Android

Большинство пользователей до сих пор думают, будто проект Android был разработан компанией Google. Но это не так. Операционная система Android была создана небольшой инновационной компанией Android Inc., расположенной в Силиконовой Долине. В июле 2005 года проект Android был приобретен компанией Google. Учредители Android Inc. работали на разные компьютерные компании, такие как Danger, Wildfire Communications, T-Mobile и WebTV. Компания Google собрала их в одну команду, и они помогли ей сделать то, что сейчас является наиболее популярной полнофункциональной операционной системой для мобильных устройств.

Совместимость с оборудованием

Операционная система Android может выполняться на устройствах разных типов с разными размерами и разрешениями экрана. Конечно, работа приложения зависит от размеров и разрешения экрана, поэтому Android поставляется с набором инструментов, помогающих создавать приложения, адаптируемые к разным параметрам экрана. Политика компании Google еще более жесткая: она принимает приложения, выполняющиеся только на совместимых устройствах. Например, если для приложения нужна фронтальная камера, в Android Market оно будет отображено только для тех типов устройств, в которые встроена фронтальная камера. Нестандартное оборудование выявляется автоматической процедурой его обнаружения. Дополнительная информация о публикации приложений в Android Market приведена в главе 8.

Устройства Android сертифицируются на совместимость на основе наличия определенного оборудования, включая следующее (список далеко не полный):

- ✓ камера;
- ✓ компас;
- ✓ GPS-приемник (Global Positioning System — система глобального позиционирования);
- ✓ трансивер Bluetooth.

Описание программы определения совместимости для специфических конфигураций устройств можно найти по адресу `http://source.android.com/compatibility/overview.html`. Сертификат совместимости гарантирует, что ваше приложение будет выполняться на всех устройствах, в списках которых оно отображено.

Мэшапы

Мэшап (mashup) — это объединение нескольких служб в одном приложении. Например, с помощью мэшапа, созданного с использованием камеры и службы идентификации местоположения, можно получить снимок и вставить в него точные географические координаты камеры в момент съемки. Комбинируя разные службы и библиотеки, можно создавать самые разнообразные приложения.

С помощью библиотек API, включенных в Android, легко комбинировать предоставляемые ими средства для создания собственного приложения. Например, можно объединить Google Maps API со списком контактов для автоматического отображения всех ваших контактов на карте.

Ниже приведено еще несколько мэшапов, которые помогут разбудить вашу фантазию. Все их компоненты, как и они сами, полностью открытые, и их можно использовать бесплатно.

- ✓ **Географическое местоположение и социальные сети.** Социальные сети сейчас очень популярны. Предположим, вы хотите создать приложение, которое передает на Твиттер ваше текущее местоположение каждые десять минут в течение дня. Сделать это очень легко. Для этого достаточно объединить в мэшапе встроенную в Android службу

идентификации местоположения с Twitter API стороннего поставщика (например, iTwitter).

- ✓ **Географическое местоположение в компьютерных играх.** Игры на основе географического местоположения игроков становятся все более популярными. Они позволяют почувствовать, что вы играете с реальным человеком, находящимся в определенном месте. В игре может выполняться фоновая служба, идентифицирующая ваше текущее местоположение и сравнивающая его с местоположением других участников игры в этом же регионе. Если в радиусе нескольких километров от вас появляется другой игрок, вы получаете извещение об этом и можете вступить с ним в контакт в качестве партнера или противника. Создать такую фоновую службу без объединения в мэшап платформы Android и технологии GPS было бы очень тяжело.
- ✓ **Интернет и контакты.** Имея в своем распоряжении мощные библиотеки Android API, легко создавать полнофункциональные приложения путем объединения нескольких библиотек. Например, можно объединить библиотеки контактов и Интернета для создания приложения, автоматически рассылающего поздравительные открытки. Можно также предоставить своим пользователям удобный способ обращения к вам из вашего же приложения или передачи этого приложения друзьям. Все это позволяют сделать встроенные программные интерфейсы.

Пусть ничто не ограничивает вашу фантазию! Эти и многие другие, еще более мощные, средства находятся буквально на вашей ладони. Вы легко можете создать приложение, записывающее географическое местоположение устройства. Платформа Android открывает доступ к огромному количеству инструментов. Их легко комбинировать любым способом для удовлетворения потребностей пользователей.

Как разработчик, вы можете делать с Android почти все, что угодно, однако соизмеряйте свою фантазию с правилами приличия и вкусами других людей. Создавая и публикуя приложения для массового использования, не забывайте о здравом смысле. Вам может нравиться анимированное фоновое изображение себя любимого на вашем дне рождения, но это не значит, что другие люди хотят смотреть на него.

Используя в приложении контактную информацию ваших пользователей для собственных маркетинговых целей, не забывайте о законодательстве, регламентирующем защиту конфиденциальной информации.

Основы программирования для Android

Чтобы создавать приложения Android, не обязательно быть семи пядей во лбу. И очень хорошо, иначе я сам не занимался бы этим. Программирование для Android — довольно простая работа, потому что для этого используется язык Java. Знание базовых элементов Java позволяет легко создавать несложные приложения Android. Конечно, для создания профессиональных приложений необходимы глубокие знания по Java, но к этому вы перейдете после прочтения данной книги.

Если вы никогда не программировали до того, как начали читать эту книгу, рекомендую ознакомиться с базовыми элементами Java с помощью какой-либо простой книги по языку, например *Java для чайников, 5-е издание*. Сейчас вам достаточно знать несколько простых конструкций Java, а именно: объявление переменных и классов, создание экземпляра класса, элементарные операторы, обработка событий и вызов методов. Изучение остальных, более сложных конструкций Java (включая наследование, инкапсуляцию и полиморфизм) можете отложить на более позднее время. После запуска своего первого приложения вам будет легче понять, что это такое.

В приложении Android почти все делается с помощью Java, однако в некоторых компонентах инфраструктуры применяются также документы XML и сценарии Apache Ant. Я рекомендую, прежде чем начать читать данную книгу, ознакомиться с базовыми элементами XML. Вам нужно знать только структуру документа XML, дескрипторы и атрибуты, все остальное пока что не нужно. Выберите самую простую книгу по XML. Сценарии Ant вам знать пока что не нужно; в примерах книги я предоставлю готовые сценарии. Эти же сценарии вы можете использовать в качестве шаблона для создания собственных приложений Android.

Если же вы хорошо знакомы с XML и Java, тем лучше. В любом случае смело приступайте к чтению книги и созданию своих первых приложений Android.

Java в приложениях Android

Приложения Android пишутся на Java, но не на полнофункциональном варианте Java, на котором профессиональные разработчики создают приложения J2EE, а на небольшом подмножестве Java, совместимом с *виртуальной машиной Dalvik*. Из подмножества, поддерживаемого Dalvik, исключены все классы, не имеющие смысла в мобильных устройствах.

Если вам что-либо не понятно в Java, найти ответ на интересующий вас вопрос можно не в учебниках, а в Google. Надо лишь удачно подобрать ключевые слова запроса, и Google выдаст сотни примеров кода Java. Язык Java существует много лет, поэтому в Интернете можно найти решение практически любой проблемы.

В виртуальную машину Dalvik включены не все пакеты и библиотеки. Создавая приложение Android, убедитесь в том, что нужный пакет доступен. Значительная часть пакетов находится в библиотеках Android.

Деятельности

Приложение Android состоит из одной или нескольких *деятельностей* (activities). Оно должно содержать как минимум одну деятельность. Пока что представляйте себе деятельность как контейнер пользовательского интерфейса, который содержит сам пользовательский интерфейс и код, создающий его. На платформе Windows аналог деятельности — форма. Более подробно деятельности рассматриваются в главах 3 и 5.

Намерения

Намерения (intents) — это сообщения на платформе Android. Намерение состоит из действия, которое нужно выполнить (просмотр, редактирование, набор номера и т.п.) и данных. *Действие* (action) — это комбинация операций, которые нужно выполнить при получении намерения, и данных, над которыми нужно выполнить указанные операции. Например, данными могут быть параметры контакта в адресной книге.

Намерения используются для запуска деятельностей и коммуникации между разными частями системы Android. Приложение может либо передавать, либо получать намерения.

Передача намерений

Когда приложение передает намерение, оно фактически посылает приказ инфраструктуре Android сделать что-либо. Например, намерение может приказать Android запустить другое приложение или новую деятельность в текущем приложении.

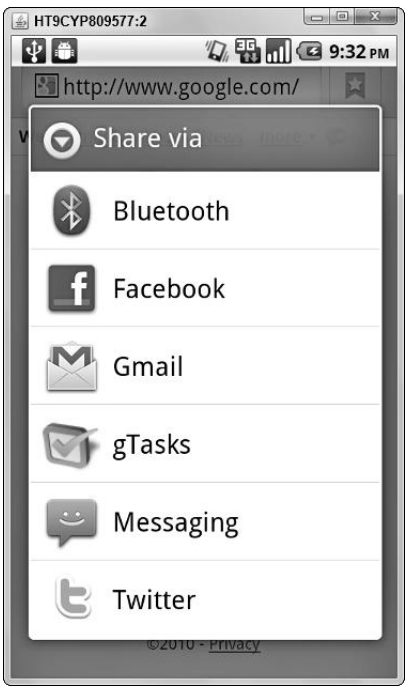


Рис. 1.1. Окно выбора приемника

Регистрация приемников намерений

Передача намерения не обязательно означает, что в приложении что-либо произойдет. Для этого нужно зарегистрировать *приемник намерений*, который прослушивает все намерения и сообщает инфраструктуре Android, что нужно сделать. Если несколько приемников могут получать одно намерение, можно создать для пользователя инструмент выбора нужного намерения. Классический пример инструмента выбора — *галерея изображений* (image gallery). Приложение отображает на экране ряд эскизов изображений и при щелчке на эскизе загружает из Интернета выбранное изображение.

По умолчанию многие зарегистрированные приемники обрабатывают общие намерения. Например, одним намерением может быть создание электронного письма, а другим — запуск другого приложения. Намерение может найти несколько приемников, поэтому пользователю предоставляется инструмент выбора (рис. 1.1), спрашивающий, что нужно сделать — создать электронное письмо или запустить другое приложение.

Если система не может автоматически найти подходящий приемник для полученного намерения, а инструмент выбора не был создан вручную, приложение генерирует исключение этапа выполнения. Необработанное исключение приводит к краху приложения. Предполагается, что разработчик знает, что он делает. Если пользовательское устройство Android не знает, как обработать переданное намерение, оно терпит крах. Поэтому рекомендуется всегда создавать инструменты выбора для намерений, нацеленных на деятельности других приложений.

Безуказательные элементы управления

В отличие от настольных компьютеров, позволяющих пользователю перемещать указатель по экрану, устройства Android позволяют выбирать команды путем прикосновения к элементу управления на экране. Но как имитировать щелчок правой кнопкой мыши для открытия контекстного меню? Вместо щелчка правой кнопкой в Android реализована концепция *длинного нажатия* (другое название — *длинный щелчок*). Прикоснитесь к значку или кнопке и не отпускайте палец несколько секунд, чтобы открыть контекстное меню. Как разработчик вы можете создать контекстное меню и задать команды, выполняемые при выборе его элементов. Кроме того, можно разрешить пользователю нажимать разные элементы двумя пальцами. Но не забывайте, что у людей могут быть пальцы разных размеров, поэтому элементы управления должны быть достаточно большими не только для миниатюрных пальчиков вашей девушки, но и для огромных, неуклюжих пальцев тракториста или грузчика. Между элементами управления должно быть большое пустое пространство, чтобы все пользователи легко могли взаимодействовать с вашим приложением.

Видовые окна и виджеты

Видовое окно — базовый элемент пользовательского интерфейса, представляющий собой прямоугольную область на экране, предназначенную для рисования и обработки событий. Фактически видовые окна являются такими же элементами управления, как и виджеты, но, в отличие от последних, они "нагружены" многими функциями. Ниже приведен ряд примеров видовых окон:

- ✓ `ContextMenu` (Контекстное меню);
- ✓ `Menu` (Меню);
- ✓ `View` (Вид);
- ✓ `SurfaceView` (Поверхность рисования).

Виджет — это элемент управления, выполняющий определенную функцию. Например, такой виджет, как флажок, определяет одно из двух возможных состояний некоторого параметра. Представляйте себе виджеты как элементы управления на экране, с которыми взаимодействует пользователь. Ниже приведены примеры виджетов:

- ✓ `Button` (Кнопка);
- ✓ `CheckBox` (Флажок);
- ✓ `DatePicker` (Инструмент выбора даты);
- ✓ `DigitalClock` (Цифровые часы);
- ✓ `Gallery` (Галерея изображений);
- ✓ `FrameLayout` (Компоновка фрейма);
- ✓ `ImageView` (Рамка изображения);
- ✓ `RelativeLayout` (Относительная компоновка);
- ✓ `PopupWindow` (Всплывающее окно).

В библиотеках Android доступны многие другие виджеты. Просмотрите содержимое пакета `android.widget` в документации Android по следующему адресу:

```
http://developer.android.com/reference/android/widget/
package-summary.html
```

Здесь вы найдете подробное описание всех виджетов, представленных в данном пакете.

Асинхронные вызовы

Класс `AsyncTask` платформы Android позволяет выполнять многие операции одновременно, избавляя от необходимости управлять отдельными потоками вручную. При запуске нового процесса с помощью класса `AsyncTask` выполняется автоматическая очистка системы, а результаты возвращаются объекту, запустившему данный процесс. Это позволяет реализовать чистую модель асинхронных вызовов.

Поток (thread) — это последовательность инструкций программы, выполняемых компьютером. Процесс — это тоже последовательность инструкций, а различие между ними состоит в том, что с каждым процессом ассоциирована изолированная область памяти, недоступная для других процессов, тогда как все потоки данного процесса имеют доступ к общей области памяти, принадлежащей данному процессу. Фактически процесс — это выполнение приложения, а потоки — одновременное выполнение многих операций в одном приложении. *Асинхронным* называется поток, выполняющийся в фоновом режиме, т.е. одновременно с другими потоками и независимо от них.

Асинхронные потоки применяются для решения длительных задач, таких, например, как загрузка большого файла, воспроизведение аудиоклипа, поиск чего-либо в Интернете и т.д. Вы не должны заставлять пользователя сидеть и ждать, пока завершится длительная задача. С помощью асинхронного потока предоставьте ему возможность заниматься в это время другими делами. Например, он может просматривать электронную почту или читать файл книги. Когда в фоновом потоке происходит что-либо важное (например, завершение загрузки большого файла), поток должен известить об этом пользователя с помощью какого-либо элемента управления. Конечно, ничто не делается само по себе. Вы, как программист, обязательно должны создать этот элемент управления, иначе пользователь, не имеющий возможности узнать, закончилась ли загрузка файла, будет очень недоволен.

Если вы не примените асинхронные потоки, пользователю покажется, что ваше приложение содержит ошибки. Пользователи привыкли к тому, что, когда приложение решает задачу, требующую некоторого времени, они могут выполнять другие операции. Если же длительная задача выполняется не в асинхронном потоке, она блокирует процессор, и устройство перестает реагировать на щелчки и нажатия. Естественно, пользователю покажется, что устройство испортилось, и он очень огорчится. Чтобы не подвергать пользователей неприятным ощущениям, в операционную систему Android встроено диалоговое окно ANR (Application Not Responding — приложение

не отвечает), показанное на рис. 1.2. Если приложение не реагирует на действия пользователя определенное время (установленное в операционной системе Android), на экране устройства отображается окно ANR. Увидев его, пользователь может решить, что делать дальше: нажать кнопку Force close (Принудительно закрыть) или Wait (Ждать). Однако не думайте, что окно ANR избавляет вас от необходимости создавать асинхронные потоки. Ваш программный продукт должен быть максимально дружественным к пользователю, иначе коммерческого успеха ему не видать.

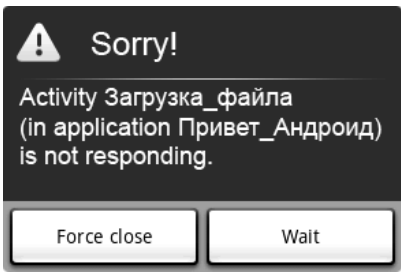


Рис. 1.2. Диалоговое окно ANR

Я рекомендую всегда запускать в отдельном фоновом потоке код, выполняющийся дольше пяти секунд и сильно загружающий процессор. Это не только мое мнение, а общепринятая методика. Обратитесь, например, на сайт разработчиков Android по следующему адресу:

```
http://developer.android.com/guide/practices/design/
responsiveness.html
```

Фоновые службы

Если у вас установлена Windows, вы, скорее всего, уже знаете, что такое *служба*. Это приложение, выполняющееся в фоновом режиме и не обязательно имеющее пользовательский интерфейс. Классический пример службы — антивирусная программа, которая обычно выполняется в фоновом режиме и отображает окно, только когда нужно что-нибудь сообщить пользователю или спросить его о чем-либо. Вы не видите ее, но знаете, что она выполняется.

Большинство аудиоплееров, доступных на Android Market, выполняется как фоновые службы. Они позволяют вам слушать музыку одновременно с проверкой почты или работой с другими задачами на экране.

Инструменты для работы с оборудованием

Компания Google предоставляет огромное количество бесплатных инструментов, необходимых для создания высококачественных мобильных приложений. Вы и другие независимые производители программного обеспечения можете использовать их, тем более что Google прилагает немалые усилия для упрощения их использования. Предоставляемые Google инструменты пригодны для всех типов оборудования.

Чтобы создать профессиональное приложение Android, необходимо задействовать все возможности оборудования. Однако не поймите меня неправильно. Если некоторая функция оборудования для разрабатываемого приложения не нужна, не пытайтесь задействовать ее лишь для того, чтобы похвастаться мощью своего приложения.

В табл. 1.1 перечислено несколько типов оборудования, которые можно использовать при создании приложения.

Таблица 1.1. Типы оборудования Android

Назначение	Оборудование
Где я нахожусь?	Устройство GPS
В каком направлении я перемещаюсь?	Встроенный компас
Как расположен мой телефон в пространстве?	Датчик положения
Мой телефон неподвижен или движется?	Акселерометр
Могу ли я использовать наушники Bluetooth?	Трансивер Bluetooth
Могу ли я записать видео?	Камера

Большинство телефонов Android поставляется с оборудованием, описанным в следующих разделах. Но не все телефоны одинаковые: операционная система Android позволяет производителям устройств включать или не включать в них определенные типы оборудования. Вряд ли существует телефон, содержащий все возможные типы оборудования.

Кроме того, по мере развития электронных технологий производители телефонов начинают добавлять средства, которые изначально не поддерживаются операционной системой Android. Однако не беспокойтесь: чаще всего производитель, добавивший в телефон новейшее уникальное средство, предоставляет набор инструментов разработки, позволяющий применять уникальное средство в приложениях Android. Например, на момент написания данной книги телефон Evo 4G компании Sprint был единственным устройством Android, обладающим встроенной фронтальной камерой. Поскольку аналогичных средств еще нет, компания Sprint выпустила пакет SDK (Software Development Kit — набор инструментов разработки), который можно использовать для доступа к уникальному новому средству. Кроме того, Sprint предоставляет образцы кодов, облегчающих реализацию нового средства.

Существуют устройства Android любых форм, размеров и типов: мобильные телефоны, планшетные компьютеры, электронные книги и т.п. В будущем появится много других типов устройств Android, таких как Google TV, а также устройств, встроенных в автомобили и бытовое оборудование. Разработчики устройств Android предоставляют инструменты, позволяющие легко разрабатывать приложения для разных типов устройств и экранов разных размеров и разрешения. Всю тяжелую работу команда Android сделала за вас. Технологии создания приложений для экранов разных размеров подробно рассматриваются в главе 4.

Сенсорный экран

Телефоны Android имеют сенсорный экран, предоставляющий огромные возможности для организации взаимодействия пользователя с вашим приложением. Прикасаясь к экрану пальцем или пальцами, пользователь может выбирать команды и пункты меню, запускать задачи, зумировать и панорамировать изображение и т.д. В приложении можно запрограммировать пользовательские жесты, что расширит его возможности.

Операционная система Android поддерживает технологию *мультитач* — управление приложением путем прикосновения к сенсорному экрану одновременно несколькими пальцами.

Традиционный элемент управления сенсорных экранов — обычная кнопка. Можно разместить кнопки любой формы в любом месте экрана, что позволяет создать максимально удобный пользовательский интерфейс.

GPS

Операционная система Android в сочетании с приемником GPS (Global Positioning System — глобальная система позиционирования), встроенным в телефон, позволяет идентифицировать местоположение пользователя в любой момент времени. Например, можно запрограммировать для родителей постоянное отслеживание перемещений их ребенка, чтобы они не беспокоились о нем. В приложении Foursquare, предназначенном для социальных сетей, GPS используется для идентификации местоположения пользователя и выяснения с помощью картографических данных, хранящихся в Интернете, в каком учреждении (ресторане, магазине, офисе, здании) пользователь находится в данный момент.

Еще одна хорошая новая возможность — определение местоположения пользователя на карте Google Map и автоматическая генерация для него инструкций, в каком направлении он должен двигаться, чтобы достичь заданной цели. Во многих приложениях Android приемник GPS применяется для подсказки пользователю, где находится ближайшая заправочная станция, кафе или гостиница. Создавая приложение Android, можно использовать библиотеки картографических функций для определения текущего местоположения пользователя на карте местности.

Акселерометр

Акселерометр — это прибор для измерения ускорения. Устройства Android постепенно учатся ощущать свое положение и перемещение в пространстве. Звучит захватывающе, не правда ли, но что нам делать с этим? Акселерометр может сообщить приложению, неподвижен мобильный телефон или перемещается, трясет его пользователь или поворачивает.

Перемещения устройства Android можно использовать в качестве входного сигнала для приложения. Предположим, вы создаете электронную игру в кости. Можно включить в приложение реакцию на встряску устройства как операцию бросания костей. Другой пример: предположим, вы создаете игру с преследованием бандитов. В этом случае можно включить уклонение игрока от бандитских пуль путем взмахов телефоном вправо или влево. Мобильное устройство становится по-настоящему мобильным: теперь его можно не только носить с собой, но и запрограммировать для него мобильные операции.

Карты памяти SD

Операционная система Android предоставляет приложениям доступ к съемным картам SD, на которых можно хранить дополнительную информацию. Начиная с версии Android 2.2 на картах SD можно не только хранить файлы, но и устанавливать приложения. Объем встроенной памяти в мобильных устройствах обычно весьма ограниченный. Это объясняется тем, что многие пользователи применяют устройство для какой-либо одной цели, например для разговоров по телефону, чтения электронных книг или работы с электронной почтой. Но если устройство используется для многих целей (игры, фотографирование, киносъемка, прослушивание музыки и др.), пользователь устанавливает дополнительные приложения, и встроенной памяти может не хватать. Отличное решение проблемы состоит в добавлении памяти путем вставки съемных карт SD. Их объем измеряется десятками гигабайтов, что позволяет пользователю загрузить из Интернета и носить с собой целую библиотеку или фонотеку.

Карты SD поддерживаются не всеми мобильными устройствами. Поэтому вставляйте в разрабатываемое приложение процедуру проверки того, есть ли в устройстве карта SD и достаточен ли объем свободной памяти. Если памяти недостаточно, приложение должно сообщить об этом пользователю.

Программные инструменты и ресурсы

Создавая приложение Android, вы имеете в своем распоряжении огромное количество готовых инструментов. В следующих разделах описаны наиболее популярные инструменты и ресурсы, которые вы будете применять при разработке приложений Android почти ежедневно.

Интернет

Благодаря встроенным в Android средствам обращения к Интернету устройство легко может получать информацию в реальном времени. В приложении можно запрограммировать мгновенное получение справки о том, какой фильм идет в соседнем кинотеатре и когда начинается очередной сеанс, когда прибывает следующий поезд, в какой гостинице есть свободные номера и т.п. Как разработчик, вы можете использовать Интернет для доступа приложения к новейшей информации, например о погоде, последних новостях, текущем счете футбольного матча. Кроме того, приложение может хранить в Интернете свои ресурсы, если в устройстве недостаточно памяти.

Можно также запрограммировать выполнение внешним веб-сервером операций, интенсивно нагружающих процессор. Это средство особенно полезно ввиду того, что в мобильных устройствах работают намного менее мощные процессоры, чем на серверах или в настольных компьютерах. В некоторых случаях передача части вычислительных операций внешним компьютерам позволяет существенно упростить приложение Android. Такое сочетание вычислительных мощностей сервера и мобильного устройства называется *клиент-серверной архитектурой*. Клиентским устройством в данном случае служит устройство Adnroid, которое передает запросы на сервер. В свою очередь сервер в любой момент времени готов

оказать клиентскому устройству требуемые услуги. Классический пример клиент-серверной архитектуры — картографические приложения, использующие ресурсы Google Maps и GPS.

Поддержка звука и видео

Операционная система Android содержит большое количество средств, облегчающих добавление звука и видео в приложение. Она поддерживает почти все стандартные форматы звука и видео. Разработчики Android сделали все возможное, чтобы добавлять мультимедийное содержимое в приложение было как можно легче. Почти без усилий с вашей стороны в приложение можно добавлять звуковые эффекты, визуальные инструкции, фоновую музыку и потоковые мультимедийные ресурсы, размещенные в Интернете. Ваша фантазия ничем не ограничена.

Список контактов

Приложение может получать доступ к контактам пользователя, хранящимся в мобильном устройстве. Можно отображать список контактов разными способами, сортировать и группировать их. Если вам не понравится стандартное встроенное приложение Contacts, можете создать собственное приложение для обработки контактов, потому что вы всегда имеете доступ к их списку. Например, можно связать в своем приложении список контактов с системой GPS, чтобы извещать пользователя о том, что он находится недалеко от точки, указанной в адресе одного из контактов.

Не ограничивайте свое воображение, но будьте ответственным. Не используйте список контактов со злонамеренными целями (см. следующий раздел).

Безопасность

Создаваемое вами приложение может делать почти все, что угодно. Однако представьте себе, что кто-то выпустил на рынок приложение, которое просматривает список контактов и втайне от пользователя передает его на сервер, принадлежащий злоумышленнику. Вот почему функции, изменяющие конфигурацию устройства пользователя или имеющие доступ к защищенному содержимому, должны иметь соответствующие разрешения. Предположим, вы хотите загрузить изображение из Интернета и сохранить его на карте SD. Для этого вам необходимо получить разрешение загружать файлы из Интернета и сохранять их на устройстве. Кроме того, необходимо получить разрешение сохранять файлы на карте SD. При установке приложения пользователь получает извещение о разрешениях, запрашиваемых приложением. В этот момент пользователь может решить, хочет ли он предоставить запрошенные разрешения и продолжить установку приложения. Добавить в приложение запрос разрешения очень легко, для этого достаточно вставить в файл манифеста приложения одну строку кода (см. главу 3).

Библиотеки Google API

Конечно, операционная система Android не ограничена телефонными звонками, организацией контактов или установкой приложений. Она предоставляет в ваше

распоряжение намного более мощные и разнообразные средства. Например, создавая приложение, можно включить в него географические карты. Для этого Google предоставляет специальные библиотеки классов, реализующие картографические виджеты.

Идентификация своего положения на карте

Предположим, вы хотите создать приложение, которое сообщает вашим друзьям о вашем текущем местоположении. Если бы для этого вам нужно было самому разработать картографическую систему, вы потратили бы на это много лет, однако существуют готовые библиотечные классы. Компания Google предоставляет вам библиотеку Android Maps API, которую можно бесплатно использовать в приложении. Это не займет у вас много времени и не будет ст*о*ить вам ни цента. С помощью Maps API приложение может найти все, что имеет адрес. Возможности безграничны: можно отобразить на экране мобильного устройства местоположение вашего друга, ближайшего овощного магазина или заправочной станции, место жительства человека, присутствующего в вашем списке контактов, и многое другое.

Путешествие по городу с помощью Android

Автоматически генерировать для друзей сообщения о своем текущем местоположении — это занятно, но Google предоставляет вам еще более мощные инструменты: библиотеку средств навигации Google Navigation API. С ее помощью можно указать точку на карте и получить подсказку, как легче всего добраться до нее с вашей текущей точки.

Принцип простоты

Создание приложения — увлекательное занятие, и неопытные разработчики часто не замечают, что код становится слишком длинным, громоздким и сложным. Поэтому никогда не забывайте о простоте. Одна из причин чрезмерного усложнения кода состоит в том, что разработчик пытается побыстрее самостоятельно решить некоторую задачу, не зная, что для ее решения существует стандартный библиотечный класс. Возможно, вы способны решить данную задачу быстрее, чем найти подходящий класс в документации Android, но учитывайте, что стандартный класс намного качественнее, чем ваш, самодельный. В поставку Android включено огромное количество библиотек классов. Вам не нужно помнить их все, но перед решением любой задачи вы должны просмотреть соответствующую категорию классов в документации Android. Вы увидите, как легко решается данная задача с помощью встроенного класса и как много времени вам сэкономит его использование. С помощью одной строки кода вы легко решите задачу, для решения которой без встроенных объектов пришлось бы писать сотни строк кода. Изменение громкости проигрывателя или создание меню — простые задачи, но, не зная классов, с помощью которых они решаются, вы засорите приложение сотнями лишних строк кода и внесете огромное количество малозаметных ошибок.

Мне это известно по собственному горькому опыту. Когда я начал создавать свое первое приложение, я поленился просмотреть скучную документацию и, как настоящий гений-самоучка, быстро написал код огромного объема, управляющий громкостью воспроизведения. Если бы я внимательнее просмотрел документацию Android, я бы нашел, что это можно было сделать с помощью одной простой строки кода. То же было и с меню. Сначала я сам создавал меню для своих приложений, при-

чем каждый раз заново. Теперь-то я уже хорошо знаю, что для этого в Android встроена стандартная инфраструктура меню, позволяющая сэкономить много времени.

Еще одна причина чрезмерного усложнения приложения — его засорение функциональностью, в которой нет реальной необходимости. Большинство пользователей хотят, чтобы операции с интерфейсом были как можно более простыми, поэтому не делайте вычурную компоновку с вкладками в случаях, когда можно обойтись несколькими пунктами меню. Не создавайте собственные элементы управления без крайней необходимости. В стандартных библиотеках Android есть огромное количество встроенных элементов управления (они называются *виджетами*), с помощью которых можно сделать почти все, что угодно. Использование встроенных виджетов облегчит жизнь не только вам, но и пользователям, потому что с вашими виджетами они не знакомы, а со стандартными уже встречались неоднократно.

Облачные вычисления

Естественно, имеются в виду не те облака, которые проплывают по небу у нас над головой. *Облако* — это специальная сетевая инфраструктура хранения и обработки информации, состоящая из множества компьютеров. В частности, облачная инфраструктура сообщений Android позволяет передавать извещения с веб-сервера в приложение. Предположим, вы сохранили все данные приложения в облаке и загрузили все ресурсы приложения из облака при первом запуске приложения. Но как быть, если после этого вы обнаружили, что один из рисунков неправильный? Приложение уже распространено среди пользователей, и вам придется обзванивать их, предлагая установить правильный рисунок. Конечно, это не реально. Нужно запрограммировать в приложении автоматическое обновление ресурсов. Чтобы приложение обновило рисунок, оно должно каким-либо образом узнать, что рисунок изменился. Для этого можно передать со своего веб-сервера облачное сообщение о том, что нужно обновить рисунок. Система облачных сообщений работает, даже если в данный момент приложение не активно или мобильное устройство выключено. Когда при следующем включении устройство получает сообщение из облака, операционная система Android запускает приложение и направляет ему сообщение. После этого приложение выполняет необходимые операции: обращается к облаку и получает новый рисунок.

Глава 2

Подготовка инструментов разработки

В этой главе...

- Как стать разработчиком приложений Android
- Инфраструктура системы разработки
- Инструменты для разработки приложений
- Инсталляция и конфигурирование инструментов разработки
- Установка JDK
- Установка Android SDK
- Установка программы Eclipse
- Знакомство с инструментами разработки Android

Все программное обеспечение, необходимое для разработки приложений Android, полностью бесплатное. Это огромное преимущество платформы Android. Жаль, конечно, что в пакете инструментов Android нельзя бесплатно получить компьютер и устанавливаемую на нем операционную систему, но вы, надеюсь, на это и не рассчитываете.

В данной главе приведены подробные инструкции по загрузке и установке инструментов и инфраструктур, необходимых для создания приложений Android.

Как стать разработчиком приложений Android

Начать разрабатывать приложения Android — несложная задача. Фактически, это намного проще, чем вам сейчас кажется. Ответьте на следующие вопросы.

- ✓ Хотите ли вы разрабатывать приложения Android?
- ✓ Хотите ли вы бесплатно получить инструменты для разработки программного обеспечения?
- ✓ Хотите ли вы бесплатно пользоваться библиотеками классов Android?
- ✓ Есть ли у вас настольный компьютер?

Если ответы на все эти вопросы положительные, значит, вы готовы стать разработчиком приложений Android. Возможно, вас удивило, что все инструменты полностью бесплатные. В наше время мало что можно получить бесплатно, и тем не менее все необходимое для разработки приложений Android можно получить, не заплатив ни копейки.

Однако где-то все же кроется подвох, верно? Так и есть. Вы можете бесплатно разрабатывать все, что вам захочется для собственного удовольствия, но как только вы

захотите опубликовать свое приложение для продажи на сайте Android Market, вам придется достать из кармана кошелек. В первую очередь вы должны будете заплатить небольшой гонорар за регистрацию приложения на сайте. На момент написания данной книги регистрация стоила 25 долларов.

Чтобы вы не начали сразу же беспокоиться насчет платежей, важно отметить, что вы можете разработать приложение для своего клиента и просто передать ему распространяемый пакет, получив от него за это некоторую плату. А ваш клиент, если вы согласны, может опубликовать приложение на сайте Android Market, используя собственную учетную запись, за которую платит он, а не вы. Так можно разрабатывать приложения и получать за них деньги, не тратя денег вообще.

Что понадобится

Теперь, когда вы готовы стать разработчиком приложений Android, сядьте за компьютер и "соберите" все необходимые для этого инструменты. Однако сначала кратко рассмотрим инфраструктуру Android, чтобы понять, что это за инструменты и зачем они нужны.

Исходные коды Android

Важно отметить, что все исходные коды Android распространяются на основе открытой лицензии (Open Source). Это означает, что вы можете не только загружать их, но и бесплатно модифицировать. Конечно, в данный момент вы вряд ли будете делать это, но, как знать, возможно, со временем вы станете профессиональным разработчиком и захотите загрузить исходные коды и создать новую версию Android или, по крайней мере, полезную для ваших пользователей модификацию существующей версии. Вам будет приятно уже сейчас узнать, что сделать это можно бесплатно. Для этого достаточно посетить хранилище Android Git. Кроме того, исходные коды можно загрузить с сайта `http://source.android.com`.

Ядро Linux 2.6

Операционная система Android основана на открытом ядре Linux 2.6. Команда Android выбрала это ядро по той причине, что оно предоставляет надежные, многократно проверенные средства управления операционной системой, необходимые для Android. Ниже перечислены наиболее важные характеристики ядра Linux 2.6.

- ✓ **Модель безопасности.** Ядро Linux надежно обеспечивает безопасность коммуникации между приложением и системой.
- ✓ **Управление памятью.** Все управление памятью выполняется автоматически, и при разработке приложения не нужно беспокоиться о размещении объектов в памяти.
- ✓ **Управление процессами.** Ресурсы предоставляются процессам автоматически по мере необходимости.
- ✓ **Сетевые стеки.** Ядро Linux поддерживает сетевое взаимодействие.

- ✓ **Модель драйверов.** Производители оборудования могут встраивать драйверы поставляемых ими устройств в Linux, чтобы обеспечить их правильную работу.

Наиболее важные средства ядра Linux 2.6 приведены на рис. 2.1.

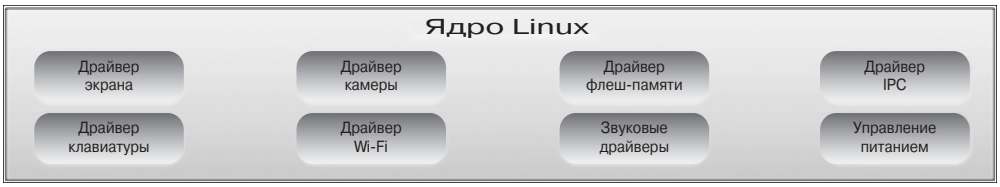


Рис. 2.1. Подмножество средств ядра Linux

Инфраструктура Android

Поверх ядра Linux 2.6 расположена инфраструктура Android, предоставляющая ряд прикладных средств. Большинство этих средств было позаимствовано из многочисленных открытых проектов. Ниже перечислены наиболее важные средства инфраструктуры Android.

- ✓ **Исполняющая среда Android.** Состоит из базовых библиотек Java и виртуальной машины Dalvik.
- ✓ **Графическая библиотека.** Используется для создания двух- и трехмерной компьютерной графики и представляет собой кроссплатформенную многоязыковую библиотеку API (Application Program Interface — программный интерфейс приложений).
- ✓ **WebKit**. Открытый движок веб-браузера, который отображает на экране содержимое веб-страниц и упрощает их загрузку.
- ✓ **SQLite.** Открытая реляционная СУБД, предназначенная для встраивания в портативные устройства.
- ✓ **Мультимедийная инфраструктура.** Набор библиотек, позволяющих записывать и воспроизводить звук и видео.
- ✓ **SSL** (Secure Sockets Layer — слой безопасных сокетов). Набор библиотек, обеспечивающих безопасность интернет-коммуникаций.

На рис. 2.2 приведены наиболее полезные библиотеки Android.

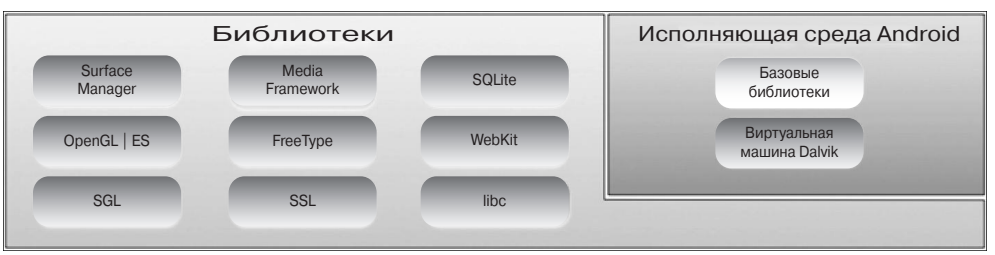


Рис. 2.2. Библиотеки Android

Инфраструктура приложения

Все библиотеки доступны разработчику приложений посредством операционной системы Android. Вам не нужно беспокоиться о том, как Android взаимодействует с базой данных SQLite или с мультимедийной инфраструктурой — они полностью готовы для использования в разрабатываемом приложении. Команда разработчиков Android выбрала для вас наиболее полезные библиотеки и предоставила их вам посредством программных интерфейсов. Эти интерфейсы служат оболочками библиотек и облегчают их использование на платформе Android. Ниже перечислены наиболее полезные библиотеки Android, которые можно использовать в приложении.

- ✓ **Activity manager** (Диспетчер деятельностей). Управляет жизненными циклами деятельностей.
- ✓ **Telephony manager** (Диспетчер телефонии). Предоставляет доступ к телефонным службам и некоторую информацию об их пользователях, например номера телефонов.
- ✓ **View system** (Система отображения). Обработка элементов управления пользовательского интерфейса.
- ✓ **Location manager** (Диспетчер расположения). Определение текущего географического местоположения устройства.

На рис. 2.3 показаны библиотеки, используемые в инфраструктуре приложений.

Рис. 2.3. Библиотеки приложений Android

Вся операционная система Android, от ядра до API, разработана на основе стабильных, проверенных временем открытых технологий, что позволяет создавать мощные приложения для мобильных устройств. Полная инфраструктура Android показана на рис. 2.4.

Иногда при разработке приложения Android возникает необходимость обратиться к ресурсам операционной системы, не имеющим программного интерфейса. Разработчики Android не рассчитывают на то, что разработчики приложений будут часто обращаться к таким ресурсам, и тем не менее это не запрещено. Вы имеете неограниченный доступ к исходным кодам Android, можете загружать и модифицировать их. Естественно, для этого нужно обладать достаточной квалификацией и учитывать, что любое изменение нужно тщательно протестировать в разных ситуациях во избежание побочных эффектов. Впрочем, исходные коды могут быть полезными не только для изменения функциональности Android, но и для того, чтобы выяснить какие-либо подробности работы библиотеки, не описанные в документации.

Рис. 2.4. Структура операционной системы Android

Библиотеки OHA

Вас не встревожило слово “альянс” в названии ассоциации OHA (Open Handset Alliance — альянс по разработке открытых стандартов мобильных устройств)? Неужели нам придется вступить в бой с могущественным коварным Дартом Вейдером на стороне сил добра? Не беспокойтесь, все не так опасно, хоть и не менее романтично и круто. Альянс — не что иное, как ряд компаний, объединивших свои усилия для достижения одной цели.

О создании ассоциации OHA было объявлено в ноябре 2007 года. На тот момент она состояла из 34 компаний. Неформальным лидером OHA была и остается Google.

На данный момент OHA состоит из 71 члена и представляет собой группу компаний — производителей оборудования и программного обеспечения, разрабатывающих инновационную стратегию развития мобильных устройств. Декларируемая цель OHA — предоставление пользователям полного перечня мощных и полезных устройств. Официальный сайт OHA имеет адрес `www.openhandsetalliance.com`.

Среди членов OHA — практически все ведущие компании, занятые поставкой оборудования и программного обеспечения для мобильных устройств, включая T-Mobile, Spring, LG, Motorola, HTC, NVidia и Texas Instruments.

Нас (автора и читателей данной книги) ассоциация OHA интересует главным образом потому, что все библиотеки операционной системы Android основаны на открытых исходных кодах. Каждый член OHA вносит свой вклад в поддержку Android. Производители чипов обеспечивают их совместимость с платформой Android,

производители оборудования отвечают за совместимость с библиотеками, другие компании вносят свой интеллектуальный вклад — коды, документацию и т.п. Общая цель всех членов OHA — коммерческий успех Android.

Члены OHA разрабатывают инновационные технологии для платформы Android. По решению OHA некоторые из этих инноваций включаются в открытый код Android, а другие остаются интеллектуальной собственностью тех членов альянса, которые разработали их.

Каждое мобильное устройство обладает собственным набором встроенного оборудования. Например, в мобильное устройство могут быть встроены GPS, Bluetooth, Wi-Fi и т.п. Наличие нужного оборудования не гарантируется. Как разработчик, вы можете уверенно полагаться только на базовую инфраструктуру Android. Для поддержки нестандартного оборудования члены OHA и другие поставщики могут предоставлять специальные библиотеки, но нет гарантии того, что эти библиотеки будут доступными на других устройствах. Впрочем, библиотеки часто разрабатываются поставщиком для конкретного (и только этого) мобильного устройства, например для электронной книги. Если устройство предназначено главным образом для предоставления конкретной функции, например для чтения книги, оно, естественно, программируется именно для этого. Реальный пример — электронная книга Barnes & Noble Nook на основе платформы Android. Она оснащена кнопками Forward (Вперед) и Back (Назад), которых нет в других устройствах Android. Следовательно, создавая программу для устройства Barnes & Noble Nook, вы запрограммируете обработчики нажатия этих кнопок, но не можете рассчитывать на то, что эти же кнопки будут присутствовать и в других устройствах Android.

Язык Java

Язык программирования Java — превосходный инструмент, существенно облегчающий программирование приложений Android по сравнению с другими мобильными платформами. Большинство других языков для мобильных устройств не обладает мощными средствами, предоставляемыми языком Java. Например, работая с другими языками, вы будете вынуждены вручную управлять памятью, размещать байты и сдвигать биты. Ввиду малой мощности процессора в мобильных устройствах традиционно использовались низкоуровневые языки, программировать на которых значительно тяжелее и сложнее, чем на высокоуровневых языках. Ситуация радикально изменилась благодаря вторжению Java в мир мобильных устройств. JVM (Java Virtual Machine — виртуальная машина Java) установлена на каждом устройстве Android и позволяет применять для программирования приложений высокоуровневый язык Java. Программисту больше не нужно вручную отображать на экране каждый пиксель. Теперь можно сосредоточиться на решении более общих задач, например на перемещении персонажа электронной игры по экрану или сортировке списка электронных книг.

Для чтения данной книги знать синтаксис Java в принципе не обязательно. Все коды на Java, необходимые для примеров книги, предоставлены читателю в готовом виде. Но для разработки приложений Android знать Java все

же необходимо. Вы, конечно, не планируете остаться на уровне "чайника" и хотите стать профессиональным программистом, поэтому я рекомендую уже сейчас найти и почитать какую-нибудь хорошую книгу по Java, например *Java для чайников, 5-е издание.*

Настройка системы

Разрабатывать приложения Android можно на компьютерах с разными операционными системами, включая Windows, Linux и Mac OS X. В данной книге предполагается, что установлена Windows 7. Если на вашем компьютере установлена другая операционная система, внешний вид некоторых окон будет немного отличаться от того, что показано в книге.

Операционная система

Среда разработки приложений Android поддерживается следующими платформами:

- ✓ Windows XP (32-разрядная), Vista (32- или 64-разрядная) и Windows 7 (32- или 64-разрядная);
- ✓ Mac OS X 10.5.8 или более поздняя версия (только на процессорах x86);
- ✓ Linux (инструменты разработки протестированы в Linux Ubuntu Hardy Heron).

Обратите внимание на то, что 64-разрядные дистрибутивы поддерживают 32-разрядные приложения.

При написании данной книги и создании рисунков для нее использовалась 64-разрядная версия Windows 7. Если на вашем компьютере установлена другая версия Windows, приведенные в книге копии экрана могут немного отличаться от того, что вы видите на экране компьютера. Если на компьютере установлена Linux или Mac, учитывайте, что маршруты файлов нужно записывать не так: `C:\path\to\file.txt`, а так: `/path/to/file.txt`.

Компьютер

Перед установкой программного обеспечения, необходимого для разработки приложений Android, убедитесь в том, что оно может адекватно выполняться на вашем компьютере. Впрочем, требования к параметрам компьютера довольно низкие. Можно считать, что подойдет любой ноутбук или настольный компьютер, выпущенный за последние четыре года. Конечно, я хотел бы более точно определить требования к компьютеру, но не могу. На момент написания данной книги официальные требования не опубликованы. Я запускал Eclipse на самом медленном компьютере, который смог достать. Это был ноутбук с процессором Pentium D, тактовой частотой 1,6 ГГц и оперативной памятью 1 Гбайт. На этом компьютере я по очереди устанавливал операционные системы Windows XP и Windows 7. В обеих операционных системах все встроенные в Eclipse инструменты выполнения и отладки работали без проблем.

Перед установкой инструментов и инфраструктур разработки приложений Android убедитесь в том, что на жестком диске достаточно свободного пространства. Требования к дисковому пространству можно найти на официальном сайте разработчиков Android:

```
http://developer.android.com/sdk/requirements.html
```

Чтобы сэкономить ваше время, я собрал собственную статистику загрузки дисковой памяти инструментами разработки приложений Android. Я обнаружил, что в наихудшей ситуации эти инструменты занимают не более 3 Гбайт дисковой памяти. Если на вашем жестком диске есть свободное место такого объема, можете устанавливать все инструменты, не читая официальные требования.

Инсталляция и конфигурирование инструментов разработки

Ниже приведен список инструментов, которые нужно установить и сконфигурировать для разработки приложений Android.

- ✓ **Java JDK.** Основа для платформы разработки.
- ✓ **Android SDK.** Предоставляет доступ к библиотекам Android и позволяет разрабатывать приложения Android.
- ✓ **Eclipse.** Интегрированная среда разработки, объединяющая Java, Android SDK и Android ADT (Android Development Tools — инструменты разработки Android). Предоставляет инструменты создания приложений Android.
- ✓ **Android ADT.** Надстройка программы Eclipse, предназначенная для решения многих задач, включая создание файлов и структур, необходимых для приложений Android.

В следующих разделах подробно рассматриваются загрузка, установка и конфигурирование каждого из этих инструментов.

Одно из огромных преимуществ открытого программного обеспечения состоит в том, что в любой момент можно загрузить необходимые инструменты и работать с ними бесплатно. Это правило в полной мере относится к инструментам разработки Android. Все инструменты, необходимые для создания мощных приложений Android, можно загрузить бесплатно. Есть и платные инструменты. Естественно, они предоставляют некоторые дополнительные услуги, но бесплатные инструменты настолько качественные, что ими пользуется большинство профессиональных разработчиков.

Установка JDK

Сначала этот пакет инструментов назывался Java SDK (Software Development Kit — набор инструментов разработки ПО), но потом компания Google решила, что краткость — сестра таланта, и переименовала его в JDK (Java Development Kit — набор инструментов разработки на Java).

Установка JDK может оказаться коварной задачей, но не беспокойтесь — я помогу вам на всех этапах загрузки и установки.

Загрузка JDK

Выполните следующие операции.

1. **Введите в адресную строку браузера адрес `http://java.sun.com/javase/downloads/index.jsp`.**

   В окне браузера будет отображена страница загрузок Java SE.

2. **Щелкните на гиперссылке `JDK` под заголовком Java Platform (JDK), как показано на рис. 2.5.**

   Данная гиперссылка указывает на страницу загрузки последнего обновления JDK (на момент написания книги — `6u26`).

Если вы работаете в Mac, установите JDK с помощью панели обновлений программного обеспечения.

Рис. 2.5. Выбор загружаемого пакета инструментов

После щелчка на гиперссылке в окне браузера откроется следующая страница загрузки Java SE, предлагающая выбрать платформу (Windows, Linux или Mac), на которой вы будете разрабатывать программное обеспечение.

3. **В раскрывающемся списке платформ выберите нужную и щелкните на кнопке Download (Загрузить).**
4. **Может появиться окно регистрации. В этом случае щелкните на гиперссылке `Skip This Step` (Пропустить этот шаг), расположенной в нижней части страницы.**
5. **В списке предлагаемых инсталляционных файлов щелкните на файле `JDK-6u26-windows-i586.exe` (если на компьютере установлена Windows).**

   Если появится диалоговое окно с предупреждением системы безопасности, закройте его. Активизируется диалоговое окно сохранения файла (рис. 2.6).
6. **Щелкните на кнопке Сохранить. Выберите папку для файла на жестком диске и еще раз щелкните на кнопке Сохранить.**

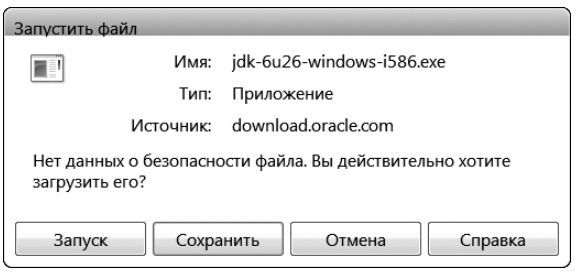


Рис. 2.6. Сохранение инсталляционного файла JDK

На момент, когда вы будете читать эту книгу, веб-страница, показанная на рис. 2.5, может выглядеть иначе. Чтобы убедиться в том, что открыта нужная страница, ознакомьтесь с требованиями к системе по адресу `http://developer.android.com/sdk/requirements.html`. Но адрес тоже может измениться. Все течет, все меняется... Однако в любом случае вы легко найдете пакет JDK. Загружать и устанавливать JRE (Java Runtime Environment — исполняющая среда Java) не нужно, поскольку она включена в JDK.

Вы должны знать, какая версия JDK вам необходима. Примеры данной книги приведены для версии Android 2.2. В Android 2.2 поддерживаются версии JDK 5 и 6. Если установить неправильную версию Java (не поддерживаемую используемой версией Android), во время разработки будут происходить неожиданные события. Впрочем, проблема решается просто: версии 5 и ниже вам не нужны. Устанавливайте версию 6, и у вас не будет с ней никаких проблем.

Инсталляция JDK

Когда загрузка инсталляционного файла будет завершена, дважды щелкните на нем, чтобы запустить установку JDK. Появится диалоговое окно, спрашивающее, разрешаете ли вы внести изменения на компьютер. Щелкните на кнопке Да (если

щелкнуть на кнопке Нет, установка будет прекращена). Прочитайте лицензионное соглашение и подтвердите согласие соблюдать его. В последнем окне щелкните на кнопке Готово. Вот и все! Пакет JDK установлен.

Установка Android SDK

В пакет Android SDK включены отладчик, библиотеки Android, эмулятор мобильных устройств, документация, образцы кодов и учебники. Разрабатывать приложения Android без Android SDK невозможно.

Загрузка Android SDK

Чтобы загрузить Android SDK, выполните следующие операции.

1. **Введите в адресной строке браузера адрес `http://developer.android.com/sdk/index.html`.**
2. **Выберите последнюю версию стартового пакета SDK для вашей операционной системы.**
3. **Сохраните стартовый пакет SDK на жестком диске.**
   Можете сохранить стартовый пакет SDK в любой папке. В данном примере предполагается, что он сохранен в папке `E:\android`.
4. **Откройте папку `android` в проводнике Windows и дважды щелкните на файле `SDK Manager.exe`** (рис. 2.7).

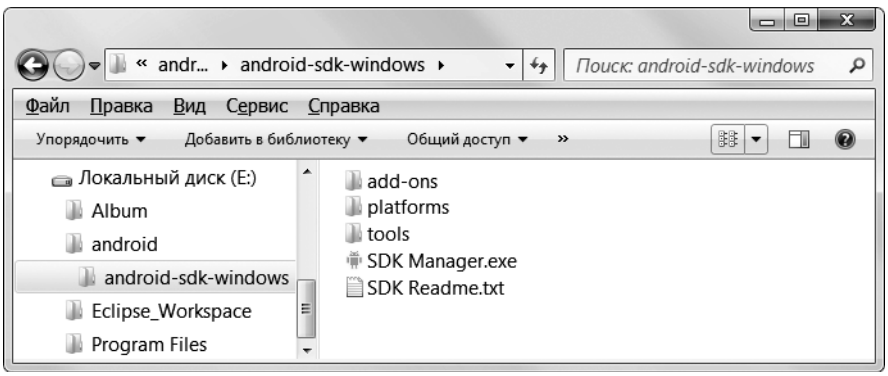


Рис. 2.7. Запуск менеджера пакетов

5. **Подтвердите подлинность файла, щелкнув на кнопке Yes (Да).**
   Активизируется диалоговое окно Android SDK and AVD Manager (Менеджер Android SDK и AVD).
6. **Выберите пакеты, показанные на рис. 2.8.**
   Сначала список появится в диалоговом окне Choose Packages to Install (Выберите устанавливаемые пакеты). Можете выбрать пакеты в нем. Но легче поступить по-другому. Щелкните на кнопке Cancel (Отмена). Вновь станет активным окно Android SDK and AVD Manager. В нем на левой панели выберите категорию Available packages (Доступные пакеты) и установите флажки напротив нужных пакетов.

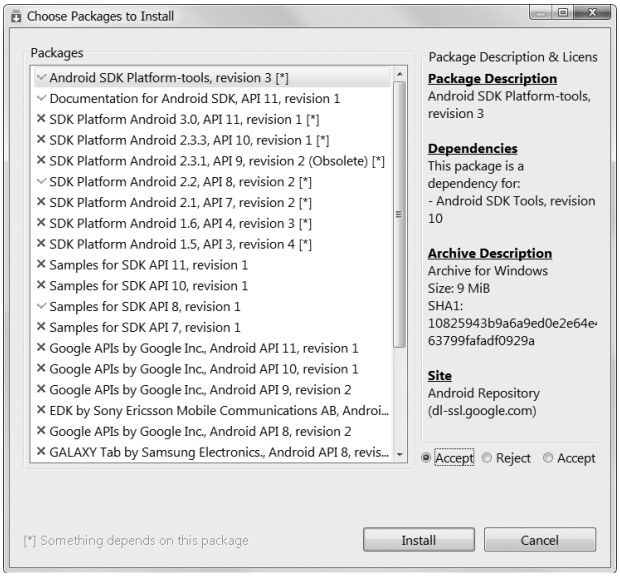


Рис. 2.8. Выбор устанавливаемых пакетов

В примерах данной книги используется версия 2.2 (см. рис. 2.8). Можно увидеть в списке и более поздние версии. Примеры книги работоспособны с ними, но пока что вам лучше точно придерживаться моих инструкций. Версии операционной системы Android 2.2 соответствует версия программного интерфейса API 8, поэтому выберите образцы кодов Samples for SDK API 8, revision 1, как показано на рис. 2.8.

Для каждой новой версии операционной системы Android компания Google предоставляет новый пакет SDK, поддерживающий функциональность, добавленную в новую версию Android. Поэтому проверьте, есть ли нужная вам функциональность в загружаемой версии. Например, если в приложении нужно использовать Bluetooth, загрузите версию Android SDK 2.0 или выше, потому что предыдущие версии не поддерживают Bluetooth.

7. **Щелкните на кнопке Install (Установить).**

   Активизируется диалоговое окно Installing Archives (Установка архивов), отображающее индикатор прогресса установки (рис. 2.9).

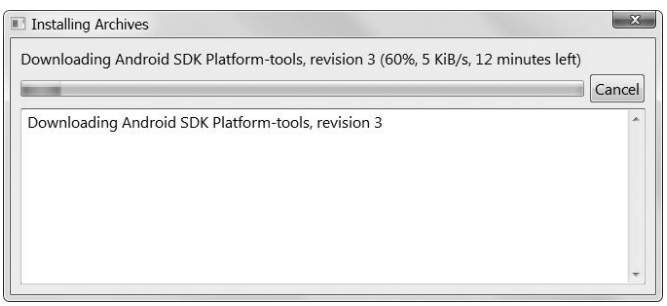


Рис. 2.9. Индикатор процесса установки пакетов

8. **Когда установка пакетов будет завершена, щелкните на кнопке Close (Закрыть).**
   В папке `android-sdk-windows` появятся вложенные папки с установленными инструментами (рис. 2.10). Их появление послужит признаком того, что вы сделали все правильно.

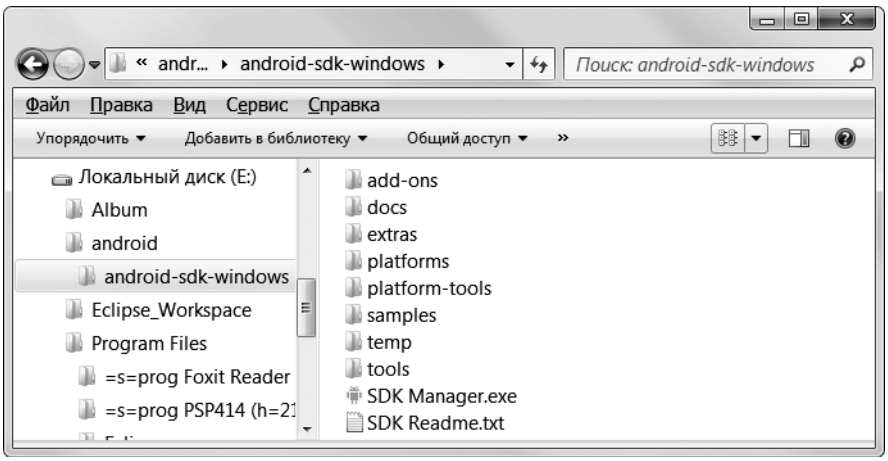


Рис. 2.10. Папки с инструментами Android SDK

Когда Android SDK пытается установить соединение с сервером для получения файлов, может быть сгенерирована ошибка `Failure to fetch URL` (Недоступен адрес URL). Возможны две причины этого: отсутствие соединения с Интернетом и неправильный протокол обмена. В первом случае решение очевидно. Если же ошибка появляется по второй причине, откройте диалоговое окно Android SDK and AVD Manager (см. выше), активизируйте на левой панели категорию Settings (Параметры) и установите флажок Force https://... sources to be fetched using http:// (Обращаться к источникам https:// с помощью протокола http://). После этого попытайтесь загрузить пакеты еще раз.

Добавление Android NDK

Пакет Android NDK (Native Development Kit — собственные инструменты разработки) представляет собой набор инструментов, позволяющих внедрять в приложение компоненты на основе собственных кодов Android, т.е. кодов, написанных на собственном языке, таком как C или C++. Если вы решите работать с NDK, вам и в этом случае нужно будет загрузить SDK. Пакет NDK не заменяет SDK, а лишь дополняет его.

Конфигурирование расположения инструментов

Этот этап не обязателен, но я настоятельно рекомендую установить маршруты инструментов Android в операционной системе Windows. Это избавит вас от необходимости всегда помнить и вводить вручную полные маршруты при обращении к инструменту ADB (Android Debug Bridge — маршрутизатор отладки Android) посредством командной строки.

Инструмент ADB позволяет управлять состоянием эмулятора устройства Android или самого устройства при отладке приложения и высокоуровневом взаимодействии с устройством. Пока что вы не будете явно обращаться к ADB; он присутствует как бы "за кулисами". Более подробную информацию о нем можно найти в документации Android.

Добавьте информацию об инструментах Android в системные переменные Windows. Для этого выполните следующие операции.

1. **Откройте панель управления Windows и щелкните на значке Система.**
2. **Щелкните на ссылке Дополнительные параметры системы (рис. 2.11). Активизируется диалоговое окно Свойства системы.**

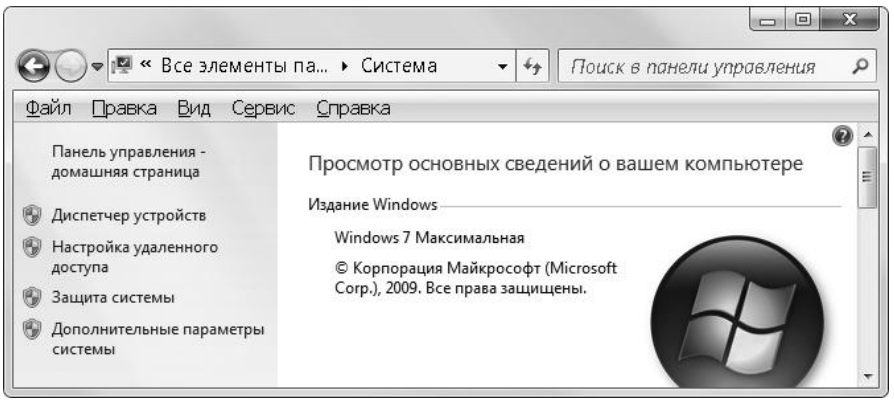


Рис. 2.11. Открытие диалогового окна Свойства системы

3. **Откройте вкладку Дополнительно и щелкните на кнопке Переменные среды (рис. 2.12). Активизируется диалоговое окно Переменные среды (рис. 2.13).**
4. **В разделе Системные переменные диалогового окна Переменные среды щелкните на кнопке Создать. Активизируется диалоговое окно Новая системная переменная.**

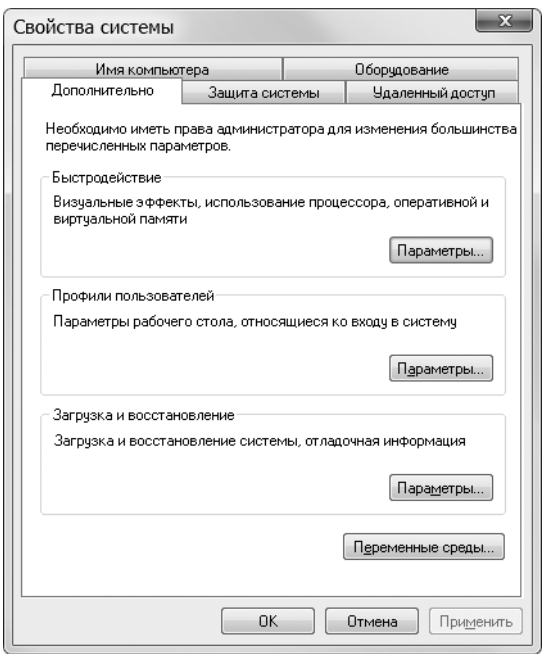


Рис. 2.12. Диалоговое окно Свойства системы

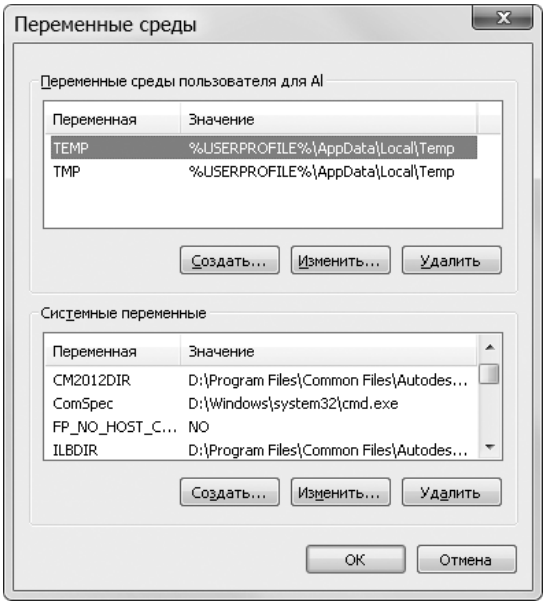


Рис. 2.13. Диалоговое окно Переменные среды

5. **В поле Имя переменной введите `ANDROID` (рис. 2.14).**
6. **В поле Значение переменной введите маршрут папки, в которой находятся инструменты Android. В данном примере предполагается, что они находятся в папке `E:\android\android-sdk-windows\tools`.**

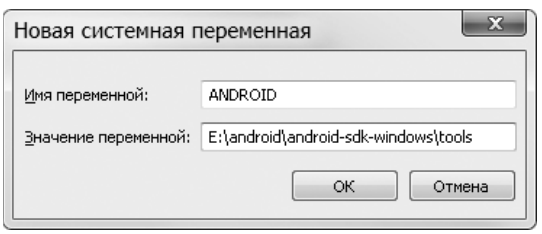


Рис. 2.14. Создание системной переменной `ANDROID`

7. **Щелкните на кнопке OK, чтобы закрыть диалоговое окно Новая системная переменная.**
8. **Добавьте маршрут инструментов Android в системную переменную `Path`. Для этого выделите ее в списке Системные переменные (рис. 2.15) и щелкните на кнопке Изменить.**
9. **Добавьте в конец значения переменной `Path` фразу `;%ANDROID%` (не забудьте добавить точку с запятой, которая разделяет маршруты).**

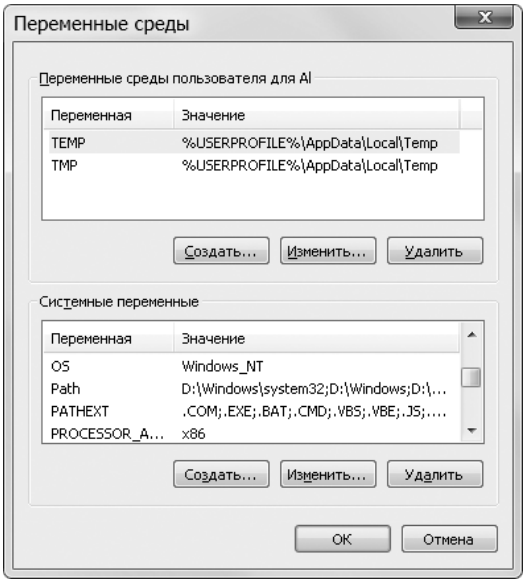


Рис. 2.15. Найдите системную переменную `Path`

Теперь при каждом обращении к папке с инструментами Android будет использоваться ее маршрут, хранящийся в системной переменной `ANDROID`.

В некоторых операционных системах изменение системной переменной не вступит в силу, пока компьютер не будет перезагружен. Поэтому проверьте ее значение. После редактирования системных переменных еще раз откройте панель управления и активизируйте диалоговое окно Переменные среды. Если системная переменная `Path` не изменилась, перезагрузите компьютер.

Установка Eclipse

Теперь, когда на компьютере установлен пакет SDK, можно приступить к установке интегрированной среды разработки Eclipse. Но сначала нужно загрузить ее из Интернета.

Выбор версии Eclipse

Важно правильно выбрать версию Eclipse. На момент написания данной книги инструменты Android не поддерживались версией Eclipse Helios (номер этой версии — 3.6). Откройте веб-страницу `http://developer.android.com/sdk/requirements.html` и посмотрите, поддерживает ли Eclipse Helios инструменты Android на момент, когда вы читаете эту книгу.

Введите в адресную строку браузера адрес `http://eclipse.org/downloads`. Откроется страница загрузки Eclipse (рис. 2.16). Выберите версию Eclipse IDE for Java Developers (Интегрированная среда разработки на Java). Расположенная рядом ссылка `Windows 32 Bit` или `...64 Bit` содержит адрес архивного файла `eclipse-java-helios-SR2-win32.zip`. Вам нужно скопировать этот файл на свой компьютер. Если у вас быстродействующее соединение и много места на жестком диске, можете загрузить версию Eclipse IDE for Java EE Developers. В нее добавлены расширенные средства для профессиональных разработчиков, которые вам пока что не нужны. Для примеров данной книги эти две версии ничем не отличаются друг от друга. Для загрузки Eclipse Galileo щелкните на ссылке `Older Versions` (Старые версии), показанной на рис. 2.16, и найдите ссылку на Eclipse Galileo.

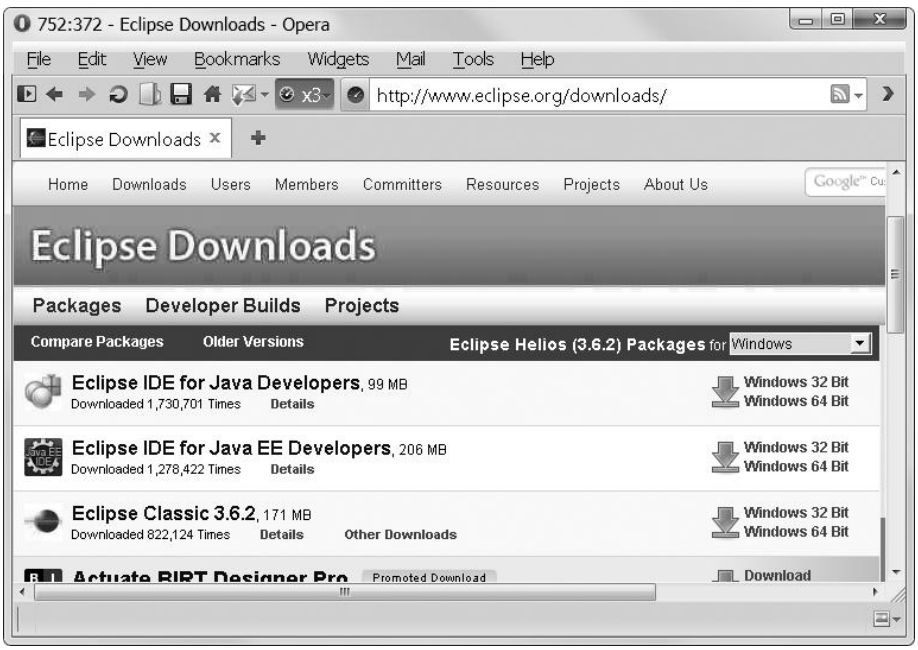


Рис. 2.16. Загрузка программы Eclipse

Инсталляция Eclipse

Программа Eclipse представляет собой самодостаточный выполняемый файл. Это означает, что для ее инсталляции нужно лишь распаковать архив `eclipse-java-helios-SR2-win32.zip` в любое место на жестком диске. В данной книге предполагается, что архив распакован в папку `E:\Program Files\Eclipse`. Чтобы с Eclipse было удобнее работать, создайте на рабочем столе ярлык файла `eclipse.exe`. Тогда вы сможете запускать программу, дважды щелкнув на ярлыке.

Выполните следующие операции.

1. **Запустите программу Eclipse, дважды щелкнув на ее ярлыке на рабочем столе.** При первом запуске Eclipse может появиться диалоговое окно системы безопасности Windows с предупреждением о том, что вы пытаетесь запустить

не сертифицированную программу. Подтвердите намерение запустить Eclipse. Для этого снимите флажок Всегда запрашивать при открытии этого файла и щелкните на кнопке Выполнить.

2. **Задайте рабочее пространство Eclipse.**

   При первом запуске Eclipse активизируется диалоговое окно Workspace Launcher (Установка рабочего пространства), показанное на рис. 2.17. Щелкнув на кнопке Browse (Обзор), можете выбрать любую папку, но в данной книге предполагается, что рабочее пространство установлено в папку `E:\Eclipse_Workspace`. Выберите папку (не обязательно указанную на рис. 2.17) и щелкните на кнопке OK.

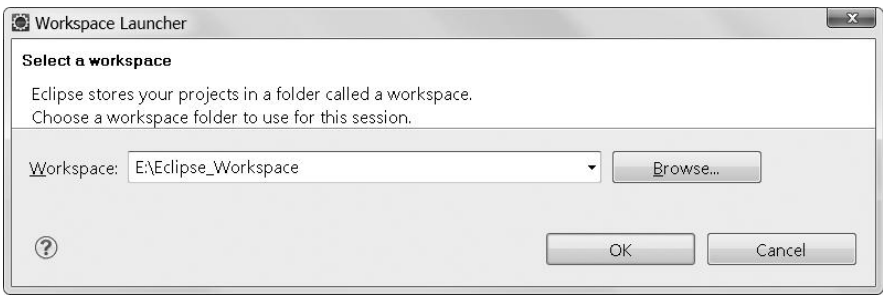


Рис. 2.17. Установка рабочего пространства

Если вы планируете работать со многими приложениями, рекомендую создать отдельное рабочее пространство для каждого проекта. При хранении нескольких проектов в одном рабочем пространстве тяжелее организовать структуру файлов и легко ошибиться, изменяя файлы с одинаковыми именами, но принадлежащие разным проектам. Кроме того, хранение проекта в отдельном рабочем пространстве облегчает поиск проекта, когда нужно вернуться к нему для исправления замеченных ошибок.

После щелчка на кнопке OK в диалоговом окне Workspace Launcher активизируется начальное окно Eclipse (рис. 2.18).

Рис. 2.18. Начальное окно Eclipse

3. **Закройте начальное окно и перейдите в среду разработки. Для этого щелкните на кнопке Workbench (Среда разработки) или на крестике в правом конце корешка вкладки Welcome (Начальное окно).**

   Сейчас программа Eclipse установлена и легко доступна. В следующем разделе рассматривается установка в Eclipse надстройки ADT, позволяющей разрабатывать приложения Android.

Конфигурирование Eclipse

Надстройка ADT (Android Development Tools — инструменты разработки Android) добавляет в Eclipse множество полезных функций, облегчающих создание приложений Android. Например, она содержит шаблоны проектов Android, позволяющие легко создавать приложения разных типов. Шаблон проекта содержит все базовые файлы приложения, и вы получаете возможность сразу начать кодирование нужной функциональности. Кроме того, надстройка ADT подключает к Eclipse средства отладки приложения, встроенные в Android SDK. И наконец, надстройка позволяет экспортировать приложение непосредственно из рабочей среды Eclipse в подписанный файл приложения в формате APK (Android Package — пакет Android), устраняя необходимость использовать командную строку для развертывания приложения. Создать файл APK можно с помощью командной строки, но это скучная работа, чреватая многими ошибками. Средства экспорта позволяют отказаться от командной строки и создать файл APK с помощью мастера, встроенного в Eclipse. Экспорт проекта в файл APK рассматривается в главе 8.

Установка надстройки ADT

Чтобы установить в Eclipse надстройку ADT, выполните следующие операции.

1. **Запустите программу Eclipse.**
2. **Выберите в меню команду Help⇨Install New Software (Справка⇨Установка нового программного обеспечения).**

   Активизируется диалоговое окно Install (рис. 2.19).
3. **Добавьте новый сайт, доступный для загрузки программного обеспечения. Для этого щелкните на кнопке Add (Добавить). Активизируется диалоговое окно Add Repository (Добавить хранилище), показанное на рис. 2.20.**

   В данном случае сайт — это адрес сервера, который хостирует хранилище программного обеспечения. Добавление сайта в Eclipse облегчает обновление надстройки ADT и другого программного обеспечения, когда появятся новые версии.
4. **В поле Name (Имя) введите имя хранилища. В данном примере предполагается, что введено имя Android ADT, но можно ввести любую другую фразу.**
5. **В поле Location (Расположение) введите адрес `https://dl-ssl.google.com/android/eclipse/`.**
6. **Щелкните на кнопке OK.**
7. Имя Android ADT будет выбрано в раскрывающемся списке Work with (Работать с), показанном на рис. 2.19. Пакеты, доступные для загрузки и установки, приведены в столбцах Name (Имя) и Version (Версия).

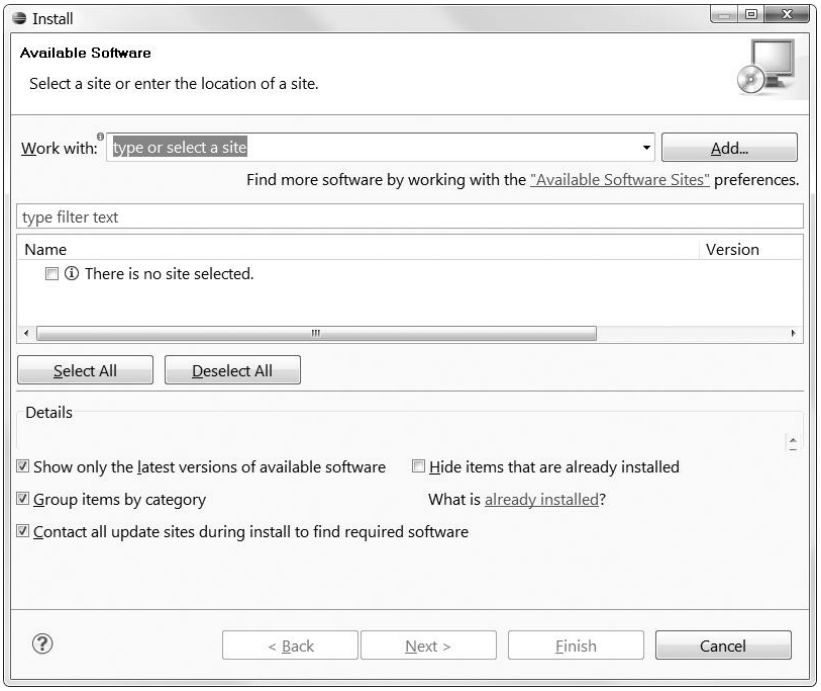


Рис. 2.19. Окно установки надстройки ADT

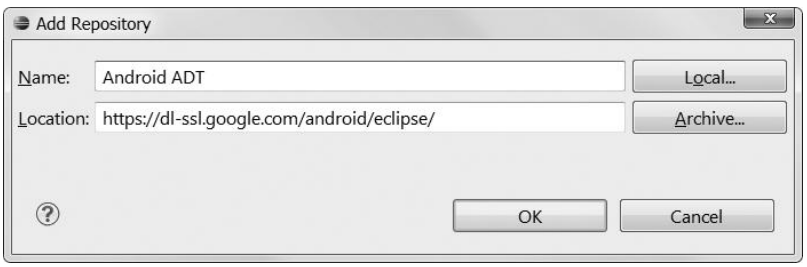


Рис. 2.20. Добавление хранилища

8. **В диалоговом окне Install установите флажки напротив пакетов, которые нужно загрузить (рис. 2.21), и щелкните на кнопке Next (Далее).**
9. **В принципе, для примеров данной книги нужны только пакеты Android DDMS и Android Development Tools, однако на всякий случай загрузите все пакеты. Они не помешают.**

   В следующем окне (рис. 2.22) приведен список выбранных вами пакетов. Щелкните на кнопке Next, чтобы начать загрузку и установку, или на кнопке Back (Назад), если хотите что-либо изменить.
10. **В следующем окне просмотрите лицензионные соглашения и щелкните на кнопке** Next.
11. **Щелкните на кнопке Finish (Готово).**

12. **Когда появится приглашение перезапустить Eclipse, щелкните на кнопке Restart Now (Перезапустить сейчас).**

Надстройка ADT установлена.

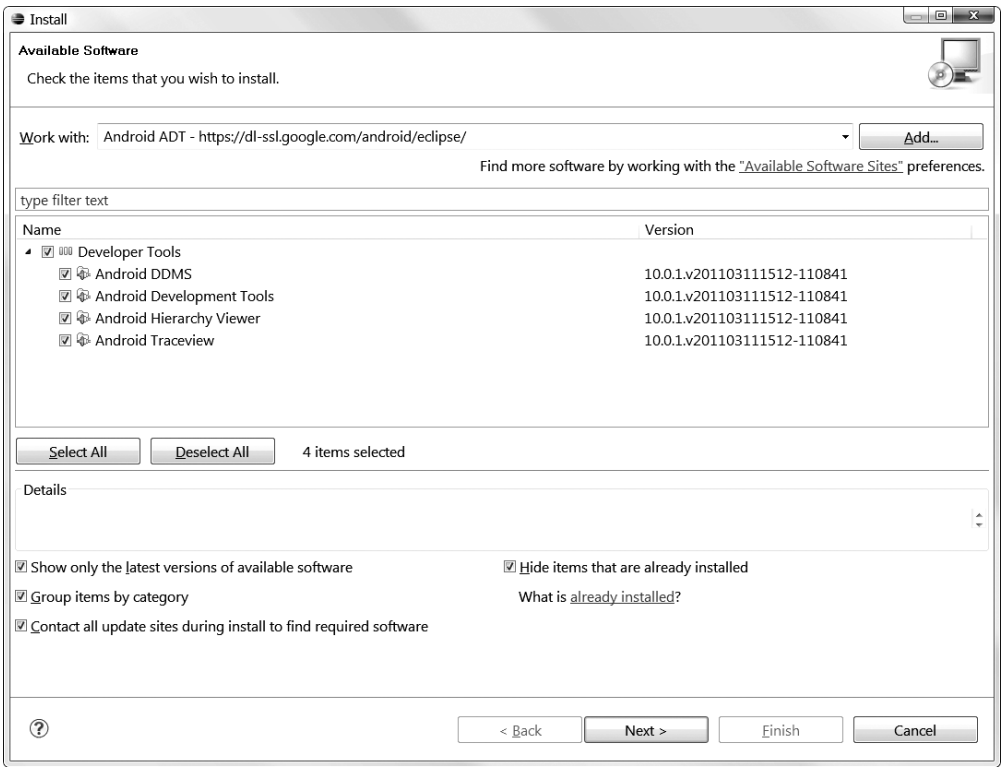


Рис. 2.21. Выбор загружаемых пакетов

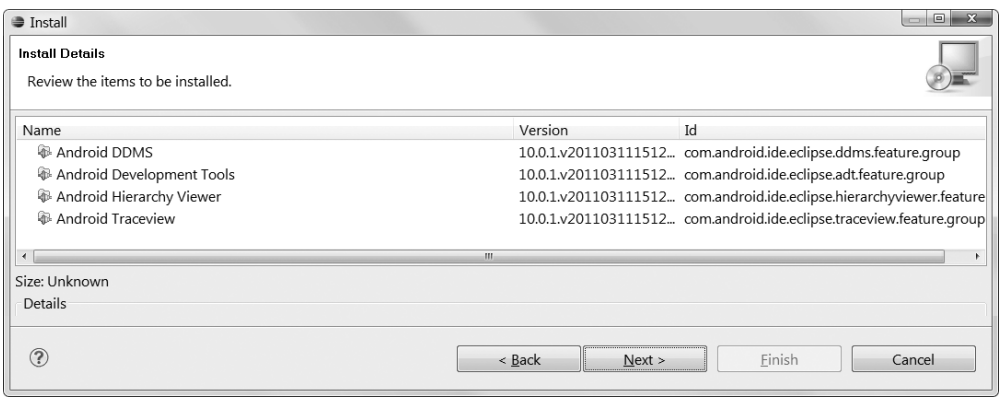


Рис. 2.22. В диалоговом окне перечислены загружаемые пакеты

Задание расположения пакета Android SDK

В этом разделе рассматривается конфигурирование маршрута Android SDK в программе Eclipse. Сделать это нужно один раз. Маршрут будет сохранен в конфигурационном файле Eclipse и будет использоваться во всех последующих сеансах работы с программой. Выполните следующие операции.

1. **Запустите Eclipse и выберите команду Window⇨Preferences (Окно⇨Параметры).**

   Активизируется диалоговое окно Preferences (рис. 2.23).
2. **На левой панели выберите категорию Android.**
3. **В поле SDK Location (Расположение SDK) введите адрес** E:\android\android-sdk-windows**.**
4. **Щелкните на кнопке OK.**

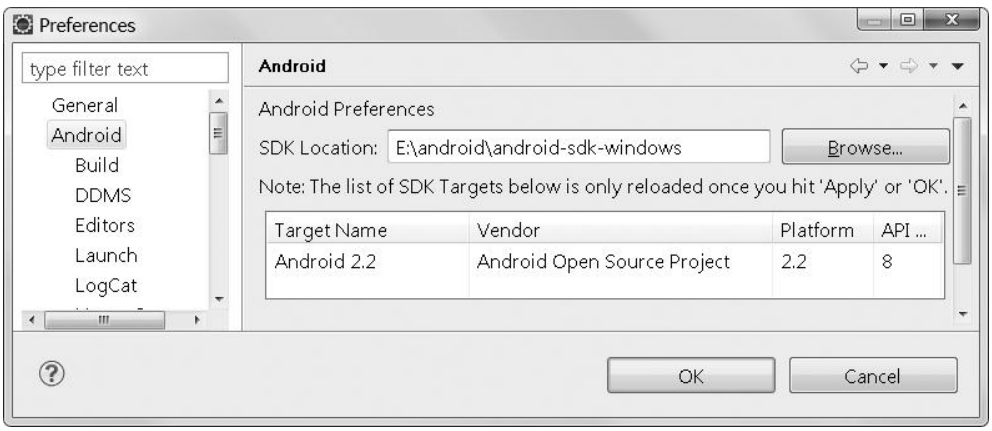


Рис. 2.23. Задание расположения пакета Android SDK

Итак, программа Eclipse сконфигурирована, и можно начинать разработку приложений Android.

Если у вас возникнут трудности при загрузке инструментов с защищенного сайта `https://dl-ssl.google.com/android/eclipse` (это может быть вызвано настройками соединения с Интернетом), попробуйте воспользоваться обычным сайтом, удалив букву `s`. Адрес будет выглядеть так: `http://dl-ssl.google.com/android/eclipse`.

Знакомство с инструментами разработки Android

Сейчас на вашем компьютере установлены и сконфигурированы все инструменты, необходимые для создания приложений Android. В данном разделе приведено краткое описание инструментов, которыми вы будете пользоваться при чтении книги и работе с примерами.

Пакет Android SDK

Откройте папку Android SDK (`E:\android\android-sdk-windows`), и вы увидите довольно много вложенных папок, содержащих разные инструменты. Вы должны хотя бы в общих чертах понимать структуру Android SDK, чтобы в полной мере воспользоваться предоставляемыми инструментами. В табл. 2.1 приведено описание каждой вложенной папки.

Таблица 2.1. Структура пакета Android SDK

Имя папки	Содержимое
`usb_driver`	Драйверы физических устройств Android. Когда устройство Android подключено к компьютеру, должен быть установлен его драйвер, иначе нельзя будет просматривать, отлаживать и развертывать приложение с помощью ADT. Папка `usb-driver` видна, когда установлен хотя бы один драйвер USB
`tools`	Все инструменты разработки, включая средства отладки, просмотра кодов, компиляции, переименования и т.п.
`temp`	Папка подкачки инструментов SDK. Иногда пакету SDK необходимо временное дисковое пространство для выполнения некоторых операций. Этой цели служит папка `temp`
`samples`	Примеры проектов, включая исходные коды и описания. Можете свободно экспериментировать с ними
`platforms`	Версии Android, для которых разрабатывается приложение. Здесь и в других местах версии называются платформами. Версия Android 2.2 называется `android-8`, версия Android 1.6 — `android-4` и т.д.
`docs`	Локальная копия документации Android SDK
`add-ons`	Библиотеки API и надстройки, предоставляющие дополнительные функции. В этой папке расположены библиотеки Google, включая картографические классы. Если не установлена ни одна надстройка, папка `add-ons` пуста

Платформы Android

Платформа Android — это всего лишь причудливое название версии Android. На момент написания данной книги были доступны семь версий, начиная с 1.1 и заканчивая 2.2. С помощью программы Eclipse можно разрабатывать приложения для любой платформы.

Учитывайте, что предыдущие версии продолжают широко использоваться в мобильных телефонах, причем пользователи не торопятся обновлять их. Если хотите, чтобы ваше приложение было доступно как можно большему количеству пользователей, выберите более низкую версию. Но выбранная версия должна быть достаточно высокой, чтобы поддерживать все средства приложения. Например, если в приложении применяется виджет Bluetooth, версия должна быть не ниже 2.0, потому что на платформах ниже 2.0 трансивер Bluetooth не поддерживается.

На рис. 2.24 показаны доли рынка для каждой платформы Android на 1 июля 2010 года. Текущую статистику для платформ можно найти по такому адресу:

```
http://developer.android.com/resources/dashboard/platform-versions.html
```

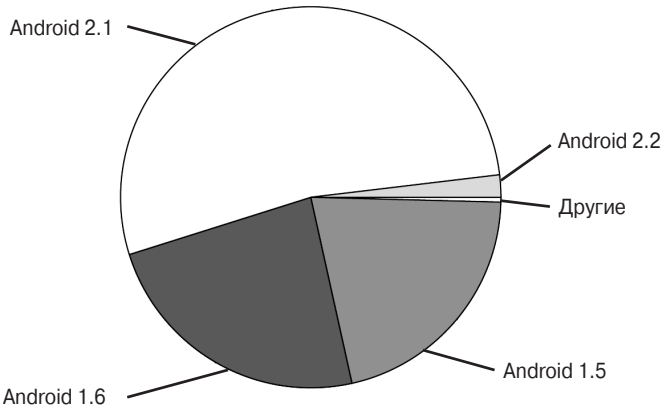


Рис. 2.24. Распространенность версий Android на 1 июля 2010 года

Использование инструментов SDK

Итак, вы установили и сконфигурировали все инструменты, необходимые для разработки приложений Android. Теперь я кратко ознакомлю вас с главными инструментами, чтобы вы могли начать работать с ними. Инструменты разработки позволяют писать код и отлаживать приложение.

Эмулятор физических устройств

Мой любимый инструмент — эмулятор. Компания Google предоставила нам не только средства разработки приложений, но и возможность тестировать их, не имея на руках физического устройства. Конечно, эмулятору присущи определенные ограничения. Например, он не может имитировать некоторые компоненты оборудования, в частности акселерометр (даже если бы акселерометр был встроен в настольный компьютер, интересно было бы посмотреть, как вы переворачиваете его вверх ногами). Но несмотря на ограничения, с помощью эмулятора можно создать и протестировать практически любое приложение.

При работе с эмулятором необходимо учитывать то, чего он не может делать физически. Например, при разработке приложения, в котором используется Bluetooth, тестирование компонента Bluetooth придется отложить до того момента, когда у вас на руках будет мобильное устройство с трансивером Bluetooth. Все остальные компоненты можно протестировать на эмуляторе устройств разных типов с разными разрешениями экрана, поэтому вам не придется покупать для тестирования все модели устройств, для которых разрабатывается приложение. Кроме того, необходимо учитывать, что на маломощном настольном компьютере эмулятор может работать медленнее, чем мобильное устройство. Когда я работаю на старом компьютере, я очень

хорошо ощущаю задержки при выполнении сравнительно простых задач, однако на новом мощном компьютере задержки не ощущаются.

Эмулятор очень удобен для тестирования приложения при разных размерах и разрешениях экрана. В Eclipse можно создать много эмуляторов с разными параметрами. Переключать приложение с одного эмулятора на другой намного легче и удобнее, чем по очереди подсоединять к компьютеру с помощью кабелей разные устройства. Не говоря уже о том, сколько вам пришлось бы потратить денег на покупку множества мобильных устройств.

Работа с реальным устройством Android

Эмулятор — прекрасный инструмент, но иногда без реального устройства все же не обойтись. Естественно, для его подключения к настольному компьютеру нужен не только кабель, но и специальное программное обеспечение — пакет DDMS (Dalvik Debug Monitor Server — сервер отладки виртуальной машины Dalvik), который вы установили в составе надстройки ADT (см. рис. 2.22). Он позволяет отлаживать приложение с использованием реального устройства, что особенно важно, когда устройство оснащено компонентами, которые невозможно эмулировать. Предположим, вы разрабатываете приложение, отслеживающее географические координаты пользователя. В эмуляторе можно установить координаты вручную, но у вас не будет полной уверенности в том, что реальное устройство правильно определяет текущие координаты. Единственный способ убедиться в этом — запустить приложение на реальном устройстве.

Если на компьютере установлена операционная система Windows, то для тестирования приложения на реальном устройстве нужно установить драйвер USB. На компьютере Mac или Linux устанавливать драйвер не нужно, поэтому можете пропустить данный раздел.

Чтобы загрузить драйвер Windows USB для устройств Android, выполните следующие операции.

1. **Запустите Eclipse и выберите в меню команду Window⇨Android SDK and AVD Manager (Окно⇨Менеджер пакетов разработки и виртуальных устройств).**
   Активизируется диалоговое окно Android SDK and AVD Manager (рис. 2.25).
2. **На левой панели выберите категорию Available Packages (Доступные пакеты).**
3. **Найдите в дереве пакетов элемент Google USB Driver package (Пакет драйвера USB компании Google), показанный на рис. 2.25. Установите флажок слева от него.**
4. **Щелкните на кнопке Install Selected (Установить выбранные).**
   Активизируется диалоговое окно Choose Packages (Выбор пакетов).
5. **Установите переключатель Accept (Принять), чтобы подтвердить согласие с условиями лицензионного соглашения, и щелкните на кнопке Install.**
   Активизируется диалоговое окно с индикатором прогресса загрузки и установки.
6. **Когда пакет будет установлен, щелкните на кнопке Close (Закрыть).**
7. **В диалоговом окне Android SDK and AVD Manager выберите на левой панели категорию Installed Packages (Установленные пакеты). В списке на правой панели должен присутствовать пакет драйвера USB (рис. 2.26). Закройте диалоговое окно Android SDK and AVD Manager.**

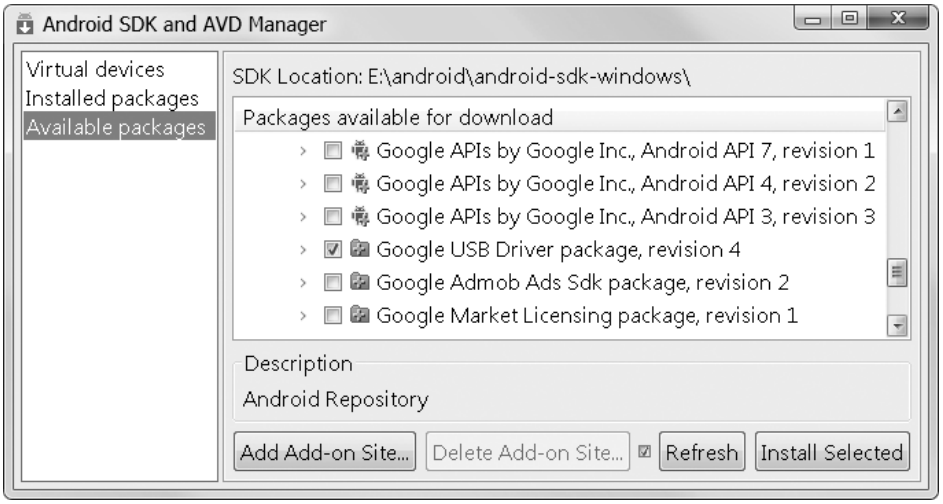


Рис. 2.25. Пакеты, доступные для загрузки

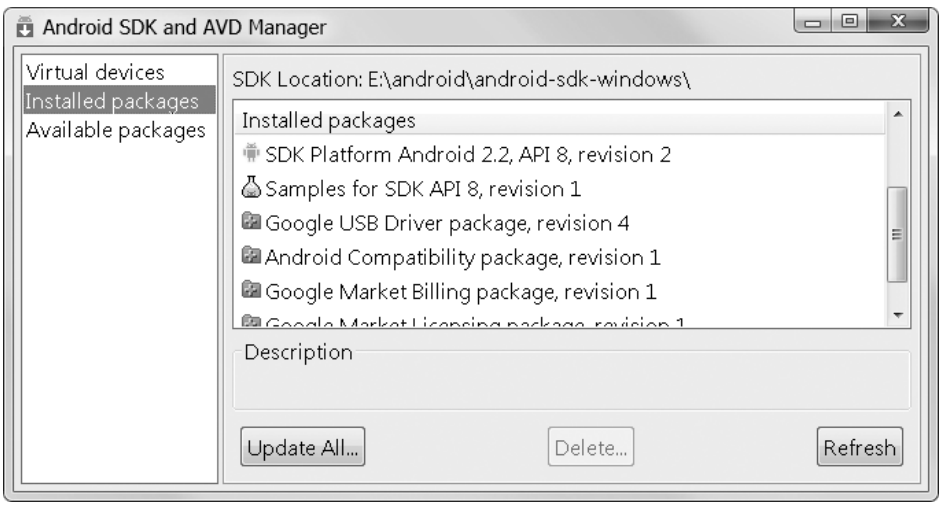


Рис. 2.26. Драйвер USB в списке установленных пакетов

Отладка приложения

Кроме драйвера USB пакет DDMS предоставляет инструменты, полезные для поиска ошибок и позволяющие контролировать состояние приложения и оборудования во время выполнения. Например, с его помощью можно определить режим беспроводного радио. Кроме того, пакет DDMS имитирует действия, которые обычно выполняются в реальном устройстве, например телефонный звонок, получение SMS или сеанс GPS. Подробное описание DDMS можно найти по такому адресу:

```
http://developer.android.com/guide/developing/tools/ddms.html
```

Примеры использования SDK и API

Для демонстрации способов использования функциональности, встроенной в API и SDK, разработчики Android создали образцы кодов, которые можно найти по такому адресу:

```
http://developer.android.com/resources/samples/index.html
```

Если у вас что-нибудь не получается или вы не можете понять, как работает некоторая функция, посетите эту страницу. На ней вы найдете коды и описания, демонстрирующие работу с любыми функциями, включая Bluetooth, двусторонний обмен текстовой информацией, двухмерные игры и т.п.

Несколько примеров есть в пакете SDK. Откройте папку `samples`, содержащую различные примеры, такие как взаимодействие со службами и манипулирование локальной базой данных. Потратьте некоторое время на эксперименты с примерами. Лучший способ изучения Android — ознакомление с работающими кодами и экспериментами с ними в среде Eclipse.

В папке `samples` приведены готовые приложения, демонстрирующие использование наиболее полезных библиотек API. Здесь показана реализация оповещений, намерений, меню, утилит поиска, предпочтений, фоновых служб и др. Дополнительные демонстрационные приложения можно бесплатно загрузить по такому адресу:

```
http://developer.android.com/resources/samples/ApiDemos/index.html
```

Часть II

Создание и публикация приложения Android

The 5th Wave

Рич Теннант

"Он очень милый, но я не могу встречаться с парнем, у которого такой рингтон".

В этой части...

Читая часть II, вы разработаете свое первое приложение Android. Сначала в этой части рассматриваются базовые инструменты создания приложений, после чего мы поговорим о разработке виджетов, с помощью которых пользователь взаимодействует с приложением. Когда приложение будет создано, я покажу вам, как добавить в него цифровую подпись и развернуть его на сайте Android Market для продажи. Заканчивается часть подробным рассмотрением вопросов, связанных с публикацией и продажей приложения.

Глава 3

Ваш первый проект Android

В этой главе...

Вам не терпится поскорее создать собственное приложение Android, которое, конечно же, будет непревзойденным шедевром и будет иметь потрясающий успех на рынке программного обеспечения? Прекрасно! Но должен честно предупредить вас, что ваше первое приложение будет всего лишь учебным примером, который я уже создал для вас. Вы не прославитесь и не заработаете деньги с его помощью, зато пройдете по всем этапам создания проекта Android и ознакомитесь с ключевыми приемами работы над приложением. Ваше первое приложение всего лишь отобразит традиционную фразу "Привет, Андроид!", причем вам не придется написать ни единой строки кода. Ни единой? Разве это возможно? Да. Благодаря встроенным в Eclipse средствам автоматической генерации кода. Минутку терпения, и я все вам покажу.

Создание проекта в Eclipse

В первую очередь нужно запустить программу Eclipse. На экране будет отображено пустое окно программы (рис. 3.1). Ни одного проекта пока что нет. Теперь все готово к разработке приложения Android.

В главе 2 рассматривалась установка в Eclipse надстройки ADT (Android Development Tools — инструменты разработки Android). В принципе, разрабатывать в Eclipse приложение Android можно и без нее, но она предоставляет ряд средств, существенно облегчающих работу. В частности, благодаря ей можно сгенерировать новое приложение Android непосредственно в меню Eclipse. Именно это вы сейчас сделаете. Чтобы создать свой первый проект Android, выполните следующие операции.

1. **Выберите в меню Eclipse команду File⇨New⇨Project (Файл⇨Новый⇨Проект).** Активизируется диалоговое окно New Project (Новый проект), показанное на рис. 3.2 и предлагающее выбрать мастер автоматической генерации проекта. Выбирая мастер, вы фактически определяете тип создаваемого проекта.
2. **Разверните узел Android.**

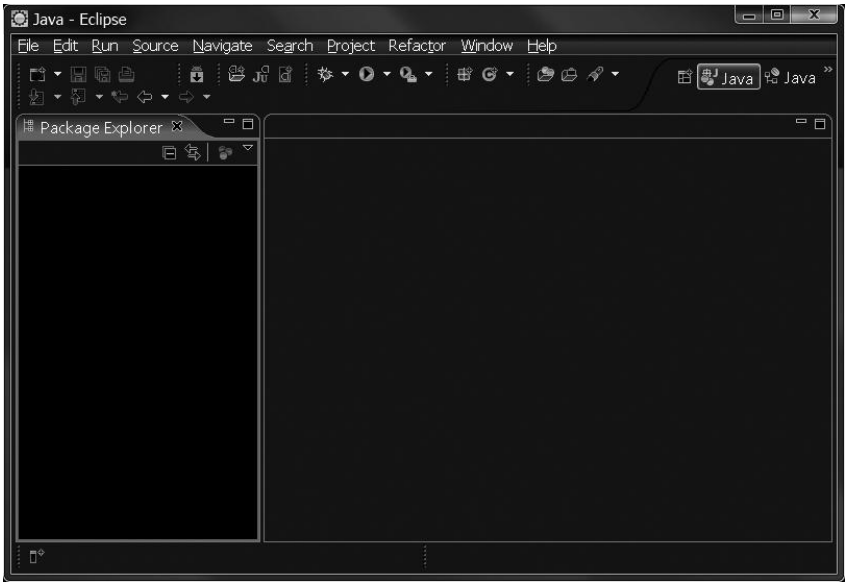


Рис. 3.1. Интегрированная среда разработки Eclipse

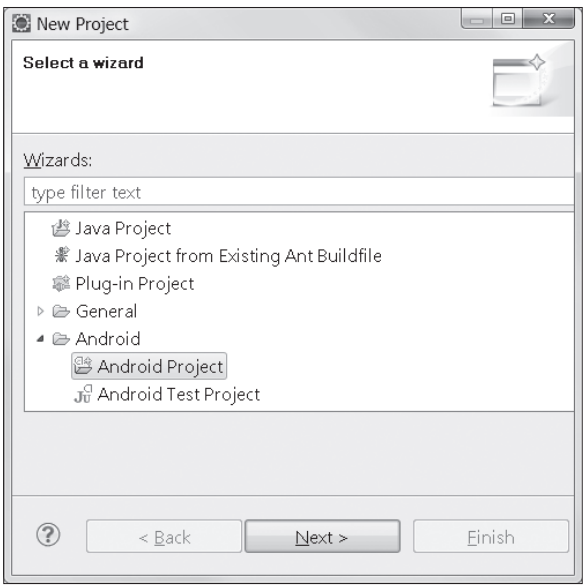


Рис. 3.2. Выбор типа создаваемого проекта

3. **Щелкните на элементе Android Project, чтобы выделить его, и щелкните на кнопке Next (Далее).**
   Активизируется диалоговое окно New Android Project (Новый проект Android), показанное на рис. 3.3.

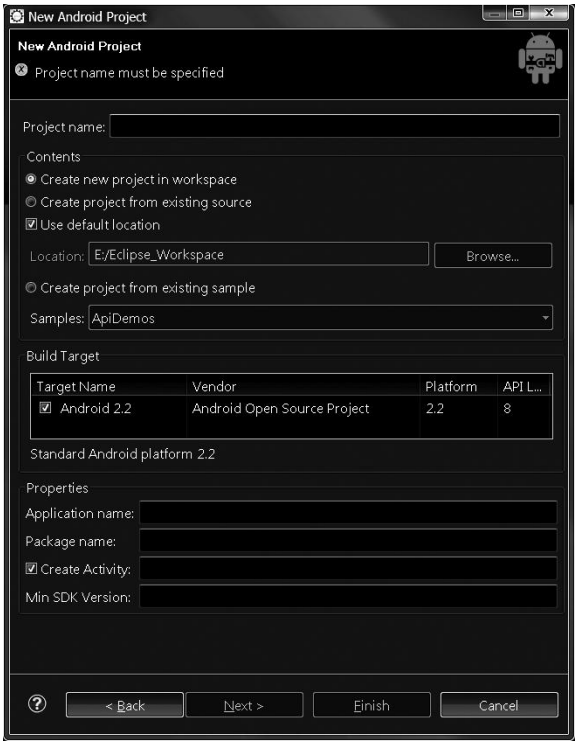


Рис. 3.3. Параметры нового проекта

4. **В поле Project Name (Имя проекта) введите фразу `Hello Android`.**
   Имя проекта — очень важный параметр. Оно идентифицирует проект в рабочем пространстве Eclipse. После создания проекта в папке рабочего пространства Eclipse будет автоматически создана папка проекта с этим именем.
5. **В разделе Contents (Содержимое) установите (или оставьте установленным) флажок Create new project in workspace (Создать новый проект в рабочем пространстве). Установите флажок Use default location (Использовать местоположение, установленное по умолчанию).**
   Маршрут папки рабочего пространства, в которой будет храниться папка проекта, вы определили в главе 2 во время конфигурирования Eclipse. Раздел Contents определяет содержимое проекта, которое будет сохранено в файловой системе. Содержимое — это исходные файлы, содержащие коды приложения Android.

После ввода имени проекта программа Eclipse создает папку с этим именем (рис. 3.4). В этой папке будут находиться файлы проекта.

Если хотите хранить файлы проекта не в папке рабочего пространства, а в другом месте, снимите флажок Use default location (Использовать местоположение, установленное по умолчанию). Станет доступным текстовое поле Location (Местоположение). Щелкните на кнопке Browse (Обзор) и найдите папку, в которой вы хотите хранить файлы проекта.

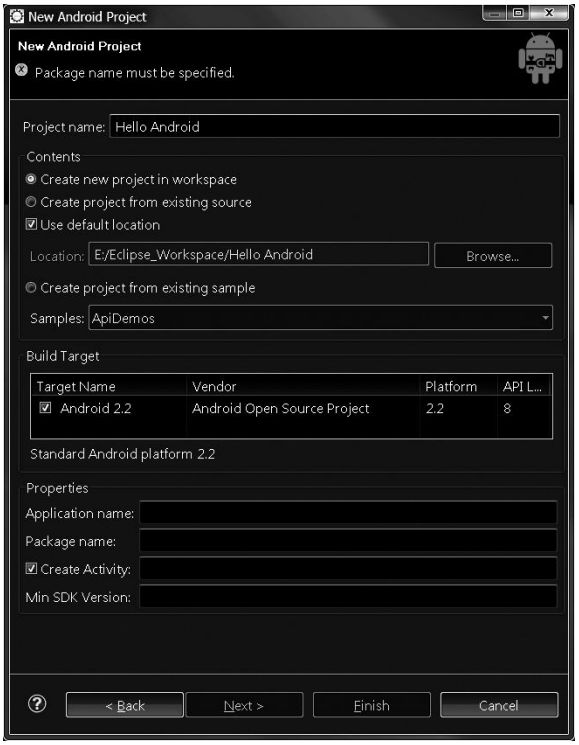


Рис. 3.4. Имя папки проекта совпадает с именем проекта

6. **В разделе Build Target (Целевая платформа компиляции) установите флажок Android 2.2.**

   Этим вы определяете программный интерфейс (т.е. фактически библиотеку классов Java), который будет использоваться разрабатываемым приложением. В главе 2 вы загрузили и установили только платформу Android 2.2, поэтому в списке присутствует только она. Для приложения будут доступны только средства, поддерживаемые целевой платформой. Например, платформа Android 2.2 предоставляет библиотеку функций распознавания устной речи, которой нет на предыдущих платформах.

7. **В поле Application Name (Имя приложения) раздела Properties (Свойства) введите `Hello Android`.**

   Это имя приложения в среде разработки Android. Оно используется в эмуляторе или физическом устройстве при установке приложения. Имена проекта и приложения не обязательно должны совпадать. В данном примере они совпадают только из-за нашей лени — чтобы не придумывать еще одно название.

8. **В поле Package Name (Имя пакета) введите `com.dummies.android.hello-android`.**

   Это имя пакета Java.

9. **В поле Create Activity (Создать деятельность) введите `MainActivity`.**

Поле Create Activity определяет деятельность, инициируемую при запуске приложения. Фактически она является точкой входа в приложение. Когда операционная система Android запускает приложение, файл этой деятельности вызывается в первую очередь. Традиционно первой деятельности приложения присваивают имя `MainActivity.java`.

10. **В поле Min SDK Version (Минимальная версия SDK) введите 8.**

    В данный момент диалоговое окно New Android Project должно выглядеть так, как показано на рис. 3.5.

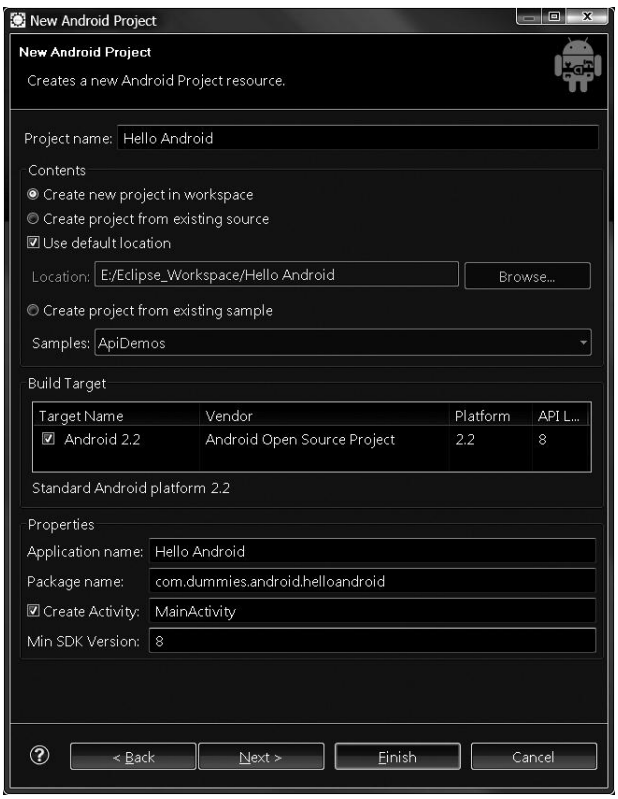


Рис. 3.5. Заполненное окно свойств создаваемого проекта

Поле Min SDK Version определяет минимальную версию платформы Android, на которой можно выполнить приложение. В принципе, это поле можно не заполнять, но тогда вы рискуете получить сообщения об ошибках на физическом устройстве после окончания отладки в среде разработки.

11. **Щелкните на кнопке Finish (Готово).**

    Созданный вами проект будет отображен в окне Package Explorer (Обозреватель пакетов) на левой панели Eclipse (рис. 3.6).

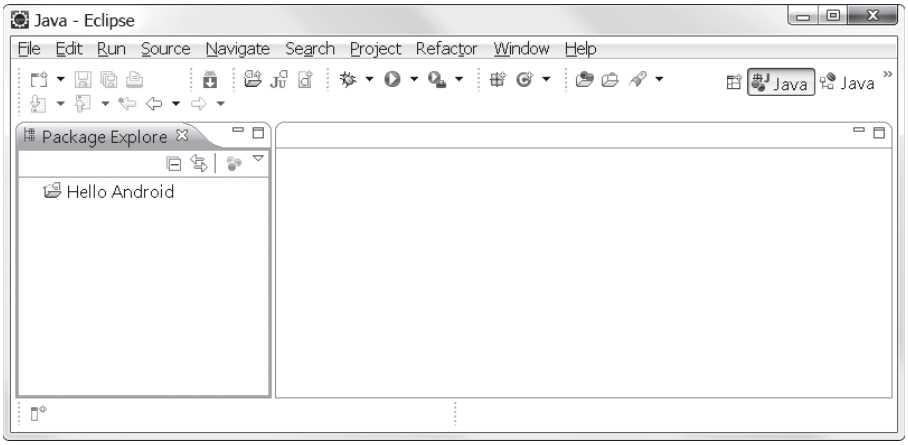


Рис. 3.6. В среде разработки появился проект `Hello Android`

Соглашение об именовании пакетов Java

Пакет — это способ группирования и упорядочения классов Java, аналог модулей и пространств имен на других платформах программирования. Каждый пакет имеет уникальное имя в рамках родительского пакета, по отношению к которому он является вложенным. Кроме того, пакет содержит классы Java. Все имена классов в каждом пакете тоже должны быть уникальными. Классы одного пакета имеют доступ к полям и методам друг друга.

Пакеты Java подчиняются соглашению об именовании на основе иерархической структуры. В полном имени пакета каждый уровень иерархии отделяется точкой. Наиболее высокий уровень — условное название организации. Последующие уровни определяются традициями и правилами, действующими в организации. В примере, рассматриваемом в данной главе, пакет называется `com.dummies.android.helloandroid`. Обратите внимание на то, что имя наиболее высокого уровня располагается в начале полного имени пакета, а вложенные уровни разделяются точками.

В пакете принято размещать классы, имеющие какое-либо отношение друг к другу. Например, в пакете можно хранить все классы, обрабатывающие веб-подключения. Тогда, во-первых, вы легко найдете нужный веб-класс, а во-вторых, классы этой категории не будут засорять другие библиотеки, не имеющие отношения к Интернету. Пакеты предназначены для упорядочения и организации кодов.

Структура проекта

В проекте Android, сгенерированном программой Eclipse, в первый момент нет скомпилированных двоичных кодов. Иногда на их создание Eclipse тратит несколько секунд, в течение которых вы можете заметить, что в системе что-то не так. Кроме того, вы должны иметь некоторое представление о высокоуровневых операциях, выполняемых “за кулисами” в Eclipse и рассматриваемых в следующих разделах.

Версии Android

Номер версии Android — это не то же самое, что номер версии программного интерфейса, применяемого в данной версии Android. Часто номер версии API называют *кодом версии Android* (не путайте с кодом версии приложения; см. далее). Между номерами и кодами версий Android существует взаимно однозначное соответствие, определяемое следующей таблицей.

Номер версии Android	Код версии
1.5	3
1.6	4
2.0	5
2.0.1	6
2.1	7
2.2	8
2.3.1	9
2.3.3	10
3.0	11

Коды версий используются в диалоговом окне New Android Project (Новый проект Android) при создании проекта и во многих диалоговых окнах при работе над проектом.

Сообщения об ошибках

После щелчка на кнопке Finish (Готово) на значке Hello Android панели Package Explorer появляется красный крестик (рис. 3.7), который исчезнет приблизительно через секунду (в зависимости от быстродействия компьютера). Таким образом рабочая среда Eclipse сообщает о том, что в рабочем пространстве проекта есть ошибки.

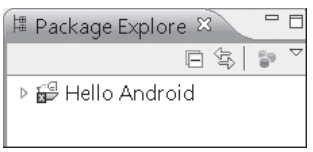


Рис. 3.7. Красный крестик сигнализирует об ошибке в проекте

По умолчанию Eclipse сконфигурирована таким образом, что при наличии ошибки в проекте на экране отображается визуальная отметка, извещающая об этом. Как могла появиться ошибка в проекте, только что созданном с помощью мастера New Android Project? Программа Eclipse и надстройка ADT выполняют некоторые операции автоматически.

- ✓ **Отображение состояния рабочего пространства.** Окно Eclipse извещает вас о существовании проблем в любом проекте, определенном в текущем рабочем пространстве. Извещением об ошибке служит красный крестик на значке проекта (см. рис. 3.7). Если проект содержит

предупреждения, вместо красного крестика отображается желтый треугольник с восклицательным знаком.

- ✓ **Автоматическая компиляция.** По умолчанию Eclipse автоматически компилирует приложение в текущем рабочем пространстве при сохранении любого файла после изменения.

Если не хотите, чтобы проект автоматически перекомпилировался после каждого изменения, снимите в меню Eclipse флажок напротив элемента Project⇨Build Automatically (Проект⇨Компилировать автоматически). Когда этот флажок снят, вы должны запускать компиляцию проекта вручную после каждого изменения исходного кода. Для этого нужно нажать клавиши <Ctrl+B>.

Так почему же появляется красный крестик после щелчка на кнопке Finish в окне создания проекта? Теперь я могу объяснить вам это явление. Когда проект добавляется в рабочее пространство, программа Eclipse и надстройка ADT проверяют, содержит ли он ошибки. В первый момент папка `gen`, содержащая сгенерированные исходные коды, еще пустая. Программа отмечает этот факт как ошибку.

Однако программа немедленно генерирует нужные коды, и папка `gen` заполняется кодами Java, которые тут же компилируются. Программа обнаруживает, что ошибки больше нет, и отметка об ошибке исчезает (рис. 3.8). Содержимое папки `gen` более подробно рассматривается далее.

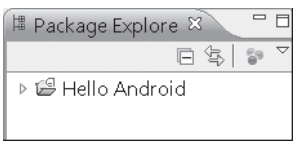


Рис. 3.8. Значок проекта извещает о том, что проект не содержит ошибок

Параметры Build Target и Min SDK Version

При создании проекта с помощью диалогового окна New Android Project вы устанавливали параметры Build Target (Целевая платформа компиляции) и Min SDK Version (Минимальная версия SDK).

Целевая платформа — это версия операционной системы Android, для которой создается код. Если выбрать платформу 2.2, будут доступны все библиотеки версии 2.2. Если выбрать 1.6, код приложения может обращаться только к библиотекам, представленным в версии 1.6. Например, при выборе версии 1.6 нельзя использовать библиотеки поддержки Bluetooth, потому что они были введены только в версии 2.0. На платформе 2.2 можно обращаться к Bluetooh API.

Нужно правильно выбрать версию до начала работы над приложением Android. Определите, какие средства Android нужны приложению, чтобы платформа поддерживала все функции, которые вы планируете использовать. Если в приложении будет использоваться Bluetooth, значит,

необходима как минимум версия 2.0. Если вы не уверены в том, какая версия поддерживает средства, которые планируете использовать в приложении, ознакомьтесь с информацией, специфичной для платформы, в разделе SDK сайта `http://d.android.com`. Например, страница платформы Android 2.2 имеет адрес `http://d.android.com/sdk/android-2.2.html`.

Коды версий и совместимость

Параметр Min SDK Version (Минимальная версия SDK) используется также сайтом Android Market (см. главу 8) для создания списка предлагаемых пользователю приложений на основе версии Android. Если на устройстве пользователя выполняется версия с кодом 3 (Android 1.6), в списке будут представлены только приложения, поддерживаемые этой версией. Сайт Android Market определяет приложения, предоставляемые каждому пользователю, на основе параметра Min SDK Version.

Если вы затрудняетесь выбрать целевую версию, принять правильное решение поможет диаграмма текущего распределения версий, приведенная на странице `http://developer.android.com/resources/dashboard/platform-versions.html`.

Анализ диаграммы распределения версий поможет вам определить, какая версия обеспечит вашему приложению максимальную долю рынка. Чем больше устройств поддерживают выбранную вами версию, тем шире будет ваша аудитория и тем больше пользователей загрузят и установят ваше приложение.

Версии операционной системы Android обладают свойством обратной совместимости. Например, если выбрать целевую версию 1.6, приложение будет выполняться на устройствах Android 1.6, 2.0, 2.1, 2.2 и всех последующих версиях (вплоть до одной из будущих версий, в которую будут внесены радикальные инфраструктурные изменения, чего, конечно, нельзя исключить). При выборе более низкой версии приложение сможет выполняться на большем количестве устройств. Однако за все нужно платить. Чем ниже выбранная версия, тем меньше функций доступны для приложения. Например, при выборе версии 1.6 в приложении нельзя использовать Bluetooth API.

Значение в поле Min SDK Version определяет минимальную версию Android, которая должна выполняться на устройстве для правильной работы данного приложения. В принципе, это поле можно не заполнять, но я настоятельно рекомендую заполнить его. Если поле Min SDK Version оставить пустым, будет применяться установленное по умолчанию значение 1. Это означает, что приложение совместимо со всеми версиями Android.

Если приложение совместимо не со всеми версиями Android (например, в нем используются библиотеки платформы 5, т.е. Android 2.0) и значение Min SDK Version не установлено, то при инсталляции приложения на устройстве с версией ниже, чем 2.0, приложение потерпит крах во время выполнения вследствие неуспешной попытки обратиться к несуществующему программному интерфейсу. Чтобы избавить пользователей от стрессов и разочарований, я рекомендую всегда устанавливать значение Min SDK Version.

Эмулятор

Чтобы на настольном компьютере увидеть, как выполняется приложение Android, нужно создать и сконфигурировать эмулятор. В рабочей среде Eclipse он называется AVD (Android Virtual Device — виртуальное устройство Android). Эмулятор выглядит и ведет себя так же, как реальное устройство, поэтому с его помощью можно тестировать приложение. Кроме того, его можно сконфигурировать на имитацию любой версии Android. Нужно лишь, чтобы соответствующая версия SDK была загружена и установлена на компьютере.

Чтобы создать эмулятор, выполните следующие операции.

1. **Откройте диалоговое окно Android SDK and AVD Manager (Менеджер пакетов разработки и виртуальных устройств Android). Для этого на панели инструментов Eclipse щелкните на кнопке, показанной на рис. 3.9.**

   Пока что ни одного эмулятора нет, поэтому список эмуляторов пустой (рис. 3.10).

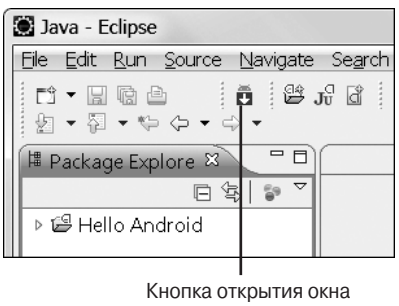


Рис. 3.9. Открытие диалогового окна Android SDK and AVD Manager

Рис. 3.10. Диалоговое окно Android SDK and AVD Manager

2. **Щелкните на кнопке New (Создать).**

   Активизируется диалоговое окно Create new Android Virtual Device (Создать виртуальное устройство Android), показанное на рис. 3.11.

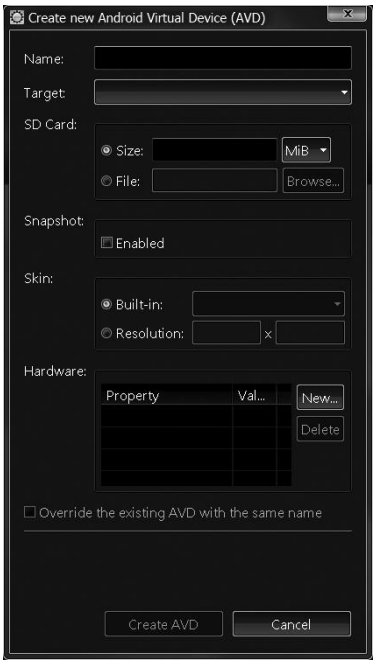


Рис. 3.11. Диалоговое окно создания эмулятора

3. **В поле Name (Имя) введите `2_2_Default_HVGA`.**

   Соглашение об именовании виртуальных устройств Android рассматривается далее.

4. **В раскрывающемся списке Target (Целевая платформа) выберите значение Android 2.2 — API Level 8.**

5. **В разделе SD Card (Карта памяти) оставьте все поля пустыми.**

   В рассматриваемом примере карта SD не используется. Если установить имитацию карты SD, для нее будет создан файл заданного размера (МБайт) в локальной файловой системе. Тогда приложение при работе в эмуляторе сможет сохранять информацию на виртуальной карте SD.

6. **В раскрывающемся списке Built-in (Встроенная) раздела Skin (Обложка) выберите значение HVGA.**

7. **В разделе Hardware (Оборудование) оставьте значения, предложенные по умолчанию.**

   В этом разделе можно задать имитацию таких устройств, как акселерометр, GSM-модем, GPS-приемник, встроенная фото- или видеокамера и т.п. Для вашего первого приложения эти устройства не нужны.

8. **Щелкните на кнопке Create AVD (Создать виртуальное устройство Android).**

   Эмулятор должен появится в списке в диалоговом окне Android SDK and AVD Manager (рис. 3.12).

9. **Закройте диалоговое окно Android SDK and AVD Manager.**

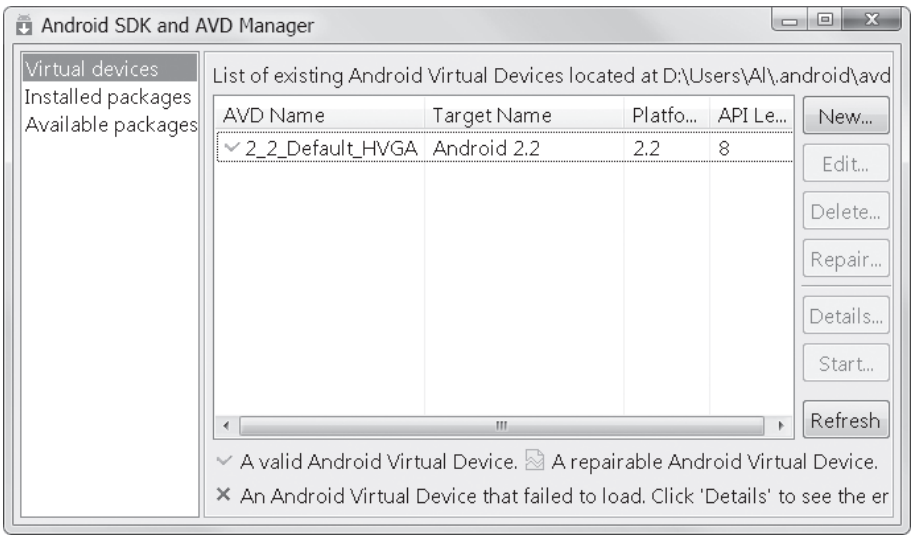


Рис. 3.12. Созданный вами эмулятор приведен в списке виртуальных устройств

После создания эмулятора может появиться следующее сообщение об ошибке: `Android requires .class compatibility set to 5.0. Please fix project properties` (Необходима совместимость классов на уровне 5.0. Исправьте свойства проекта.). В этом случае щелкните в Eclipse правой кнопкой мыши на проекте и выберите в контекстном меню команду Android Tools⇨Fix Project Properties (Инструменты Android⇨Исправление свойств проекта). Программа Eclipse автоматически исправит ошибку, не открывая никаких окон и не требуя от вас никаких действий.

Соглашение об именовании виртуальных устройств

Необходимо аккуратно выбрать имя для создаваемого виртуального устройства. Операционная система Android установлена на многих мобильных устройствах, таких как телефоны, электронные книги, планшеты и т.п. Как разработчик, вы должны будете тестировать приложения при разных конфигурациях и обзавестись многими эмуляторами, поэтому соблюдение соглашения об именовании виртуальных устройств поможет в будущем различать их. При создании имени эмулятора рекомендуется придерживаться следующего синтаксиса:

`{`*`целевая_версия`*`}_{`*`обложка`*`}_{`*`экран`*`}[{_`*`параметры`*`}]`

В приведенном выше примере задается имя `2_2_Default_HVGA`. Оно означает целевую версию Android 2.2. Номер версии 2.2 записан как `2_2`. Символ подчеркивания используется вместо точки для того, чтобы имя эмулятора было единым идентификатором. В будущем это позволит использовать имя эмулятора в сценариях командной строки.

Часть *обложка* содержит имя обложки, используемой в эмуляторе и определяющей внешний вид устройства. Обложка `Default` (По умолчанию) предоставляется пакетом Android SDK.

Часть *экран* обозначает тип графической карты и (не обязательно) разрешение экрана. При необходимости в имени эмулятора можно указать разрешение экрана, например `WVVGA800`.

Конфигурирование параметров запуска приложения

Мы почти подошли к моменту, когда вы наконец-то сможете запустить приложение на выполнение. Но для этого нужно сначала сообщить программе Eclipse, что и как вы хотите запустить. Конфигурация выполнения задает применяемый проект, запускаемую деятельность, эмулятор, на котором должно выполняться приложение, и ряд других параметров запуска. Все это нужно установить один раз, после чего надстройка ADT будет автоматически выполнять все необходимые операции при каждом запуске.

Надстройка ADT предоставляет два варианта конфигураций запуска.

- ✓ **Конфигурация выполнения.** Используется для выполнения приложения на заданном устройстве. В процессе разработки приложения Android эта конфигурация используется почти все время.
- ✓ **Конфигурация отладки.** Применяется для отладки приложения, выполняющегося на заданном устройстве.

Когда вы впервые выполняете проект как приложение Android, выбрав команду Run⇨Run (Выполнить⇨Выполнить), надстройка ADT автоматически создает конфигурацию выполнения, которая называется New_configuration (Новая конфигурация). Она не обязательно правильная, потому что Eclipse может не угадать, что и как вы хотите запустить. Если вам повезет и приложение запустится, эта конфигурация будет применяться при каждом выборе команды Run⇨Run. В противном случае (если приложение не запустится), создайте конфигурацию выполнения вручную, как описано в следующих разделах.

Создание конфигурации отладки

Пока что мы не будем заниматься отладкой приложения, поэтому отметим лишь, что для создания конфигурации отладки нужно выбрать команду Run⇨Debug Configurations (Выполнить⇨Конфигурации отладки) и сконфигурировать процесс отладки в появившемся диалоговом окне.

Создание конфигурации выполнения

Конечно, вы можете положиться на конфигурацию выполнения, созданную по умолчанию, но если вы амбициозны, создайте собственную конфигурацию, которая будет полезной во многих ситуациях. Для этого выполните следующие операции.

1. **Выберите команду Run⇨Run Configurations (Выполнить⇨Конфигурации выполнения).**

   Активизируется диалоговое окно Run Configurations (рис. 3.13). На левой панели представлен длинный список разных типов конфигураций выполнения. Нас интересуют только два следующих типа:

   - Android Application (Приложение Android);
   - Android JUnit Test (Тест Android JUnit).

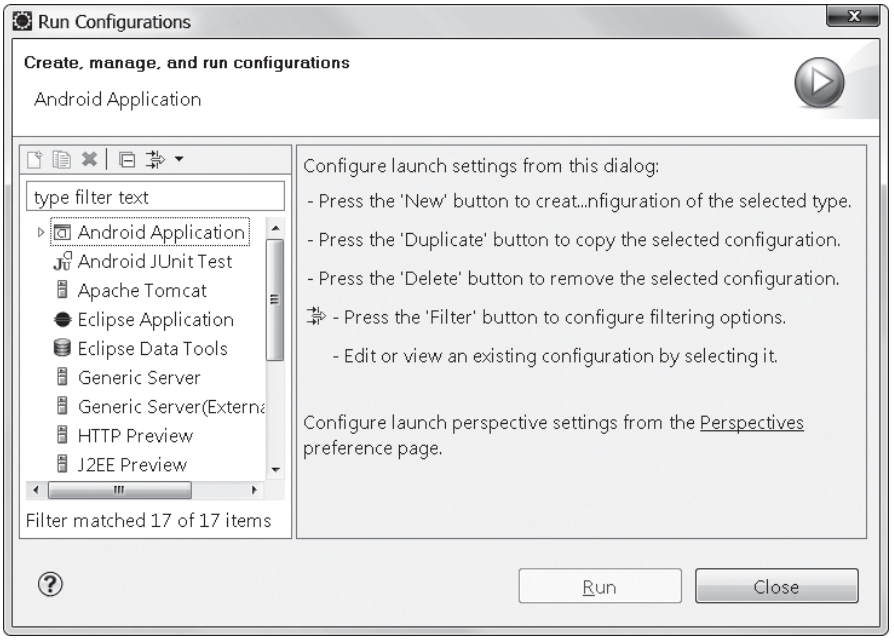


Рис. 3.13. Диалоговое окно Run Configurations

2. **Щелкните на элементе Android Application, чтобы выделить его, и на кнопке New Launch Configuration (Новая конфигурация запуска), показанной на рис. 3.14. Можно также щелкнуть правой кнопкой мыши на элементе Android Application и выбрать в контекстном меню команду New (Создать).**
   На правой панели появятся вкладки, содержащие параметры новой конфигурации.

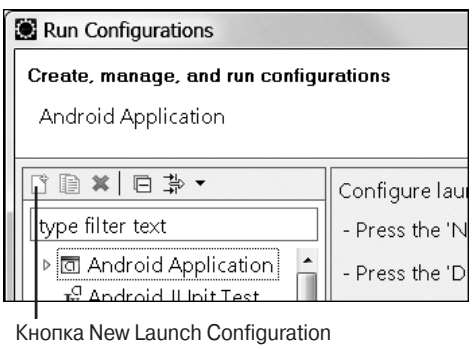


Рис. 3.14. Открытие вкладок создания конфигурации

3. **Активизируйте вкладку Android. В поле Name (Имя) введите `ExampleConfiguration`. Это имя создаваемой конфигурации.**
4. **Щелкните на кнопке Browse (Обзор).**
   Активизируется диалоговое окно Project Selection (Выбор проекта).

5. **Выберите проект Hello Android (рис. 3.15) и щелкните на кнопке OK.**
   Диалоговое окно Project Selection закроется, и вкладка Android вновь станет активной.

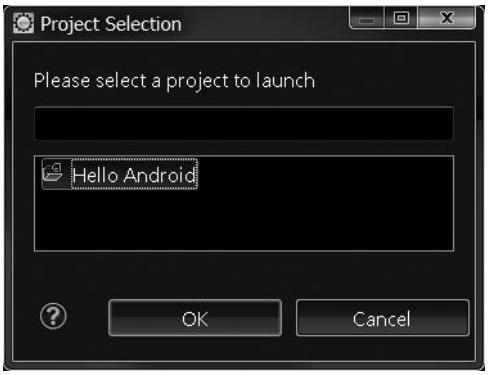


Рис. 3.15. Выбор проекта

6. **Оставьте установленным переключатель Launch Default Activity (Запустить деятельность, установленную по умолчанию).**
   Деятельность, задаваемую по умолчанию, традиционно принято называть `MainActivity`. В нашем примере она так и называется (см. выше).
7. **Активизируйте вкладку Target (Целевая платформа), показанную на рис. 3.16. Оставьте установленным переключатель Automatic.**
8. **Обратите внимание на то, что в столбце** AVD Name (Имя виртуального устройства) присутствует созданный вами эмулятор 2_2_Default_HVGA.

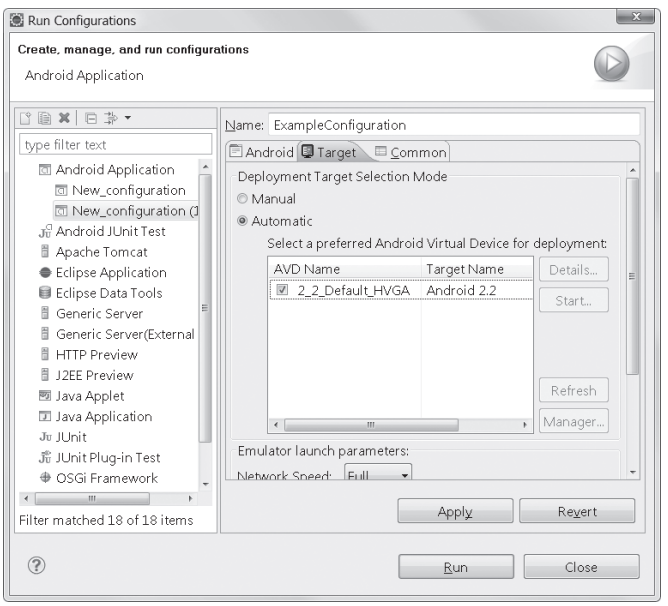


Рис. 3.16. Вкладка Target

9. **Установите флажок напротив эмулятора 2_2_Default_HVGA.**

   Установив этот флажок, вы задаете запуск приложения в созданном вами эмуляторе при выборе команды Run⇨Run. Если установить переключатель Manual (Вручную), то при каждом запуске приложения с данной конфигурацией выполнения вам будет предложено выбрать устройство, на котором должно быть выполнено приложение. Когда установлен переключатель Automatic, применяется устройство, заданное в этой вкладке (в данном примере — эмулятор 2_2_Default_HVGA).

10. **Оставьте остальные параметры установленными по умолчанию и щелкните на кнопках Apply (Применить) и Close (Закрыть).**

Теперь конфигурация ExampleConfiguration должна присутствовать в списке конфигураций на левой панели диалогового окна Run Configurations.

Дублирование конфигураций запуска

На определенном этапе вам обязательно придется проверять, как выполняется приложение на разных устройствах. Конфигурации выполнения позволяют легко и быстро запускать приложение в разных средах, но создание конфигураций может оказаться трудоемкой задачей, особенно если их много и они незначительно отличаются друг от друга. К счастью, надстройка ADT позволяет легко дублировать существующие конфигурации, благодаря чему можно быстро создавать большое количество вариантов конфигураций и устанавливать их параметры независимо от других вариантов.

Чтобы продублировать существующую конфигурацию, выполните следующие операции.

1. **Активизируйте диалоговое окно конфигураций выполнения.**

   Для этого выберите команду Run⇨Run Configurations (Выполнить⇨Конфигурации выполнения).

2. **На левой панели щелкните правой кнопкой мыши на конфигурации** ExampleConfiguration**, которую вы создали в предыдущем упражнении. Выберите в контекстном меню команду Duplicate (Продублировать).**

   Будет создана точная копия выбранной конфигурации с именем ExampleConfiguration (1).

3. **Измените имя конфигурации. Для этого введите в поле Name (Имя) фразу** `DuplicateTest`**.**

   Сейчас новая конфигурация существует, и можно изменить ее параметры (вам ведь не нужны две совершенно одинаковые конфигурации выполнения).

   Впрочем, вам сейчас не нужен и дубликат. Мы создали его только для демонстрации. Поэтому удалите его.

4. **Чтобы удалить конфигурацию DuplicateTest, выделите ее имя на левой панели и щелкните на кнопке Delete (Удалить), расположенной на панели инструментов. Можете также щелкнуть правой кнопкой мыши на имени конфигурации, приведенном на левой панели, и выбрать в контекстном меню команду Delete.**

5. **Щелкните на кнопке Close, чтобы закрыть диалоговое окно Run Configurations.**

Выполнение приложения

Мы создали эмулятор и подробно рассмотрели конфигурирование параметров запуска приложения. Теперь наступило время запустить приложение в эмуляторе. Наконец-то!

Выполнение приложения в эмуляторе

Запустить приложение очень просто. Для этого достаточно выбрать команду Run⇨Run (Выполнить⇨Выполнить) или нажать клавиши <Ctrl+F11>. Согласно вашим инструкциям надстройка ADT запустит эмулятор, созданный вами ранее, скомпилирует приложение и развернет его в эмуляторе.

Если конфигурации запуска не существует (например, вы забыли создать ее), будет открыто диалоговое окно Run As (Выполнить как), показанное на рис. 3.17. Выберите в нем пункт Android Application (Приложение Android), и конфигурация запуска будет создана автоматически.

Рис. 3.17. Диалоговое окно Run As

Если же вы создали конфигурацию выполнения ExampleConfiguration, как описано выше, будет загружен эмулятор (рис. 3.18).

При первом запуске эмулятора надпись ANDROID присутствует на экране более десяти минут! Наверное, вы уже подумали, что система зависла, но не беспокойтесь! При следующих запусках приложения в этом же экземпляре эмулятора оно будет загружаться не более чем за секунду. Объясняется это тем, что в эмуляторе выполняется виртуальная операционная система Linux. Перед первым запуском ее нужно загрузить и инициализировать. Чем медленнее компьютер, тем дольше загружается Linux и запускается эмулятор.

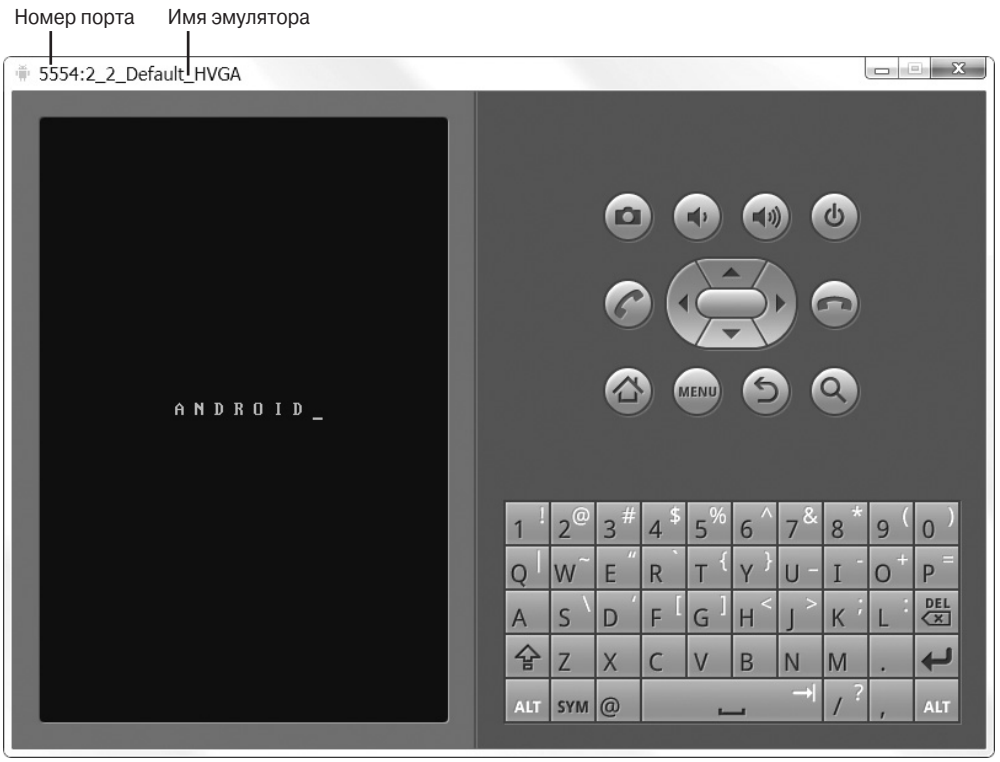


Рис. 3.18. Начальное состояние эмулятора непосредственно после запуска приложения

При запуске эмулятора по очереди отображаются несколько *загрузочных экранов*. Первый был показан на рис. 3.18. В строке заголовка окна эмулятора приведены номер порта, через который эмулятор подключен к компьютеру, и имя эмулятора. Первый экран эмулятора присутствует на экране компьютера приблизительно половину времени загрузки.

На втором загрузочном экране нарисован логотип Android (рис. 3.19). Этот же логотип пользователь видит на экране своего мобильного устройства, когда оно загружается после включения. Впрочем, некоторые производители мобильных устройств устанавливают отображение своего логотипа.

На третьем, окончательном загрузочном экране (рис. 3.20) представлен полностью загруженный эмулятор.

Чтобы сэкономить свое время, не закрывайте окно эмулятора после окончания сеанса приложения. Эмулятор не обязательно загружать заново при каждом запуске приложения. Когда эмулятор работает, можете изменить исходный код проекта и заново запустить приложение, выбрав команду Run⇨Run. Надстройка ADT найдет работающий эмулятор и развернет в нем приложение в течение секунды.

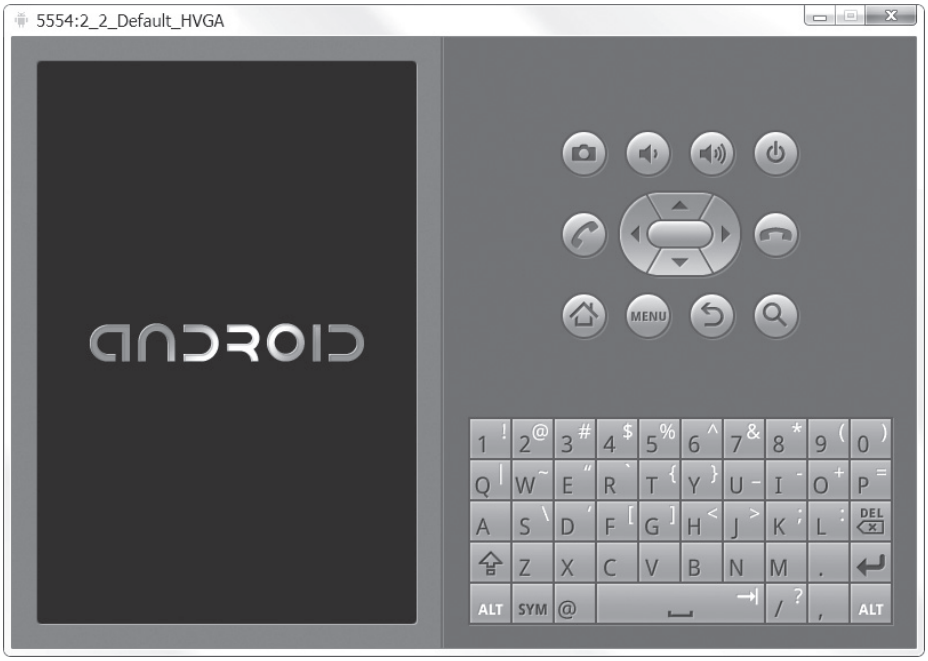


Рис. 3.19. Логотип Android на втором загрузочном экране

Рис. 3.20. Загруженный эмулятор 2_2_Default_HVGA

Когда загрузка эмулятора завершается, по умолчанию отображается заблокированный главный экран. В мобильном телефоне блокировка экрана — полезное средство безопасности, но кто украдет эмулятор с экрана вашего компьютера? Только не подумайте, будто создатели эмулятора добавили блокировку, чтобы эмулятор не украли. Она нужна для того, чтобы в целях отладки приложения эмулятор работал точно так же, как реальное устройство. Чтобы разблокировать главный экран, перетащите вправо значок Lock (Блокировка), на котором изображен замок. В правом конце экрана отпустите кнопку мыши. Во время перетаскивания фон значка изменится на зеленый, а надпись — на Unlock (Разблокировано), как показано на рис. 3.21.

Рис. 3.21. Разблокировка главного экрана

Главный экран после разблокировки показан на рис. 3.22.

Чтобы запустить приложение Hello Android, выведите на экран список приложений. Для этого щелкните на кнопке, показанной на рис. 3.22. В списке приложений щелкните на значке Hello Android. Приложение запустится и отобразит на экране надпись, показанную на рис. 3.23. Фраза на русском языке (Привет, Андроид) вставлена в файл `strings.xml` вместо фразы Hello, Android. Если после разблокировки немного подождать, приложение запустится само по себе, без щелчка на значке приложения, поскольку команда Run⇨Run заставляет запустить текущий проект. Список приложений полезен, если вы перейдете к другим приложениям, а потом захотите вернуться к приложению Hello Android.

Рис. 3.22. Разблокированный главный экран

Рис. 3.23. Приложение Hello Android выполняется и отображает фразу `Привет, Андроид...`

Если после запуска эмулятора не разблокировать экран, надстройка ADT не запустит приложение. Если после разблокировки приложение не запускается в течение нескольких секунд, перейдите в рабочую среду Eclipse (не закрывая эмулятор) и вновь выберите команду Run⇨Run. Приложение будет повторно развернуто в эмуляторе и начнет выполняться. Все выполняемые при этом операции можно увидеть во вкладке Console (Консоль) в рабочей среде Eclipse (рис. 3.24).

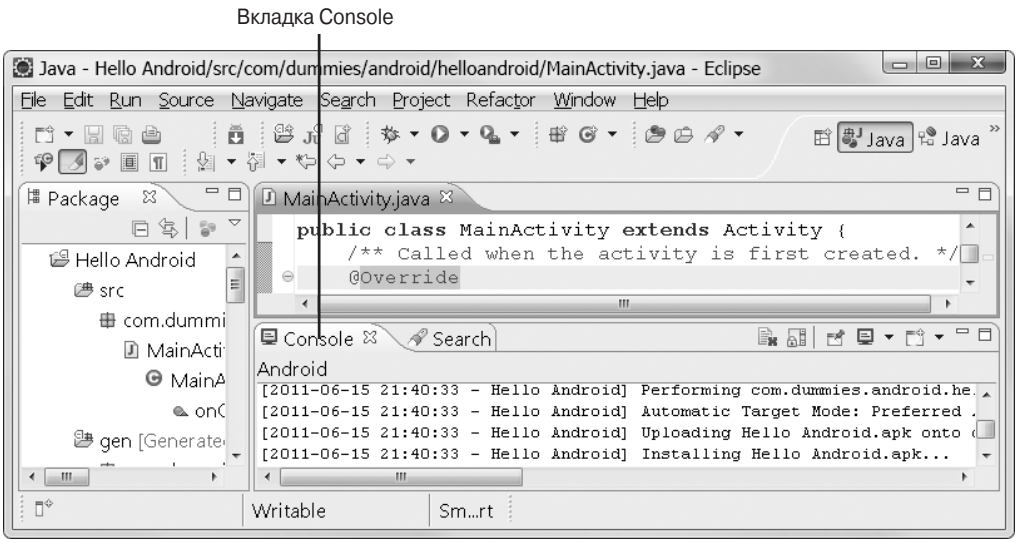


Рис. 3.24. Список операций, выполняемых при запуске приложения в эмуляторе

Информация о статусе развертывания

Во вкладке Console можно наблюдать состояние приложения в каждый момент времени при его развертывании в эмуляторе. Ниже приведен полный текст вкладки Console при развертывании приложения Hello Android.

```
[2011-07-05 13:13:46 - Hello Android] ---------------------
[2011-07-05 13:13:46 - Hello Android] Android Launch!
[2011-07-05 13:13:46 - Hello Android] adb running normally.
[2011-07-05 13:13:46 - Hello Android] Performing
              com.dummies.android.helloandroid.MainActivity
              activity launch
[2011-07-05 13:13:46 - Hello Android] Automatic Target
              Mode: using existing emulator 'emulator-5554'
              running compatible AVD '2_2_Default_HVGA'
[2011-07-05 13:13:48 - Hello Android] Application already
              deployed. No need to reinstall.
[2011-07-05 13:13:48 - Hello Android] Starting activity
              com.dummies.android.helloandroid.MainActivity
              on device
[2011-07-05 13:13:49 - Hello Android] ActivityManager:
              Starting: Intent {act=android.intent.action.
```

```
                MAIN cat=[android.intent.category.LAUNCHER]
                cmp=com.dummies.android.helloandroid/
               .MainActivity }
[2011-07-05 13:13:49 - Hello Android] ActivityManager:
                Warning: Activity not started, its current
                task has been brought to the front
```

Вкладка Console предоставляет ценную информацию о процессе развертывания приложения. Глядя на записи, можно узнать, когда запускается деятельность, на каком устройстве надстройка развертывает приложение и пр. В последних трех строках данного примера выводится следующее предупреждение.

```
[2011-07-05 13:13:49 - Hello Android] ActivityManager:
                Warning: Activity not started, its current
                task has been brought to the front
```

Надстройка ADT предупреждает, что деятельность `MainActivity` не запустилась, потому что она уже выполняется. Поскольку деятельность активна, надстройка отображает ее на переднем плане экрана Android, в результате чего пользователь видит ее.

Папка проекта

Итак, вы создали свое первое приложение. Вы сделали это, не написав ни единой строки кода. Надстройка ADT предоставляет инструменты, которые позволяют быстро создать несложное приложение, ничего не зная ни о Java, ни о структуре проекта. Однако при создании более сложного приложения вы должны понимать структуру проекта, созданную с помощью диалогового окна New Android Project (Новый проект Android). Начиная с этого момента вы часто будете работать с файлами проекта.

Рекомендую внимательно прочитать следующие разделы, потому что на протяжении всей своей карьеры разработчика приложений для Android вы бо́льшую часть времени будете проводить, исследуя папки и файлы проекта. Понимание их структуры и назначения — ключ к успешной работе над сложными приложениями.

Папки приложения

На рис. 3.25 показан проект Hello Android в окне Package Explorer (Обозреватель пакетов).

Когда проект развернут, на левой панели отображаются следующие папки:

- ✓ `src`;
- ✓ `gen`;
- ✓ `Android 2.2`;
- ✓ `assets`;
- ✓ `res`.

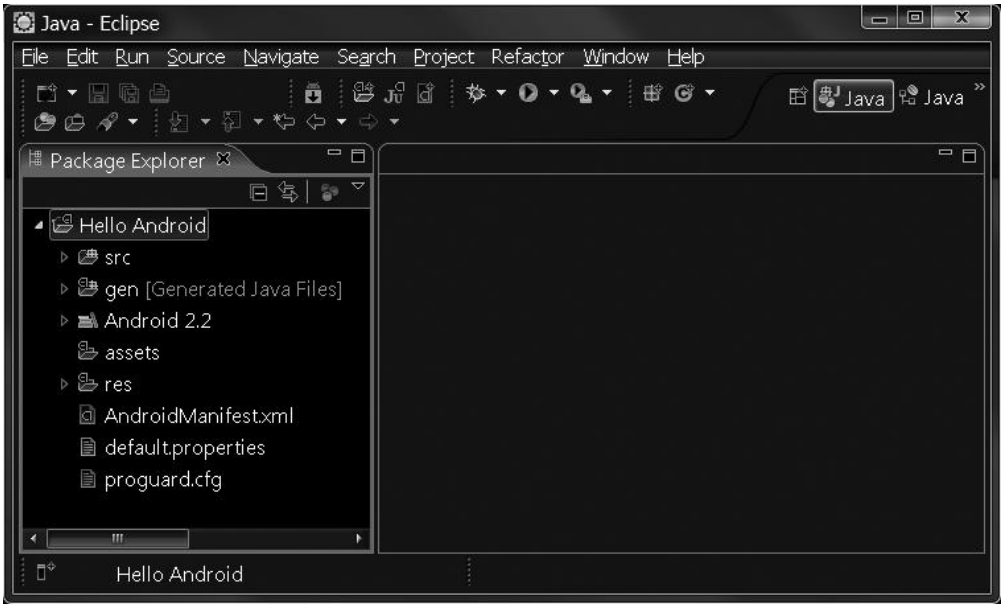


Рис. 3.25. Структура проекта Hello Android

В более сложных проектах могут использоваться и другие папки, такие как `bin`, `resources`, `libs`, `Referenced Libraries` и т.д. На рис. 3.25 отображены папки, сгенерированные по умолчанию мастером New Android Project (Новый проект Android).

Сначала папка `bin` не видна, потому что в последних версиях ADT в окне обозревателя проектов она скрыта (возможно, в будущих версиях ADT она опять будет отображена). Папок `libs` и `Referenced Libraries` не существует, пока вы не добавите в проект библиотеки сторонних поставщиков и ссылки на другие библиотеки (см. далее).

В проекте используются файлы `AndroidManifest.xml` и `default.properties`. Файл `AndroidManifest.xml` идентифицирует компоненты, которые компилируют и выполняют приложение, а файл `default.properties` содержит свойства проекта Android, установленные по умолчанию (например, версию Android). Содержимое и назначение этих папок и файлов рассматриваются в следующих разделах.

Папка `src`

Эта папка содержит исходные коды проекта Android, включая главный файл `MainActivity.java`, созданный мастером New Android Project (Новый проект Android) ранее. Чтобы увидеть содержимое папки `src`, нужно развернуть ее. Для этого выполните следующие операции.

1. **Щелкните на стрелочке слева от имени папки `src`.**
   Узел `src` будет развернут, и вы увидите в нем вложенную папку пакета `com.dummies.android.helloandroid`.
2. **Разверните пакет, щелкнув на стрелочке слева.**
   В пакете вы увидите файл `MainActivity.java` (рис. 3.26).

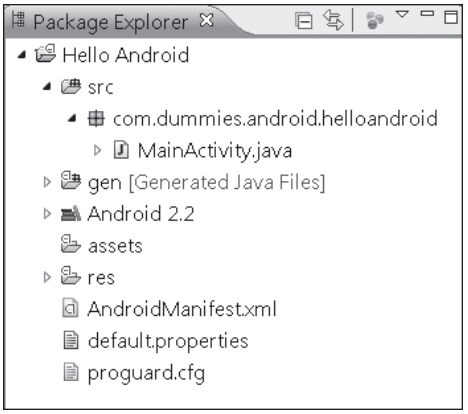


Рис. 3.26. Содержимое папки `src`

Если хотите, можете создать несколько пакетов. Можете также присваивать пакетам простые имена типа `p1`, `p2` и т.д. Однако учитывайте, что пакет служит именем пространства имен, поэтому очень желательно, чтобы его имя уникально идентифицировало приложение. Тогда вы сможете смело подключать к приложению любые библиотеки, не боясь, что имена сущностей в приложении будут конфликтовать с именами в библиотеках. Каждый пакет содержит несколько классов приложения. Для каждого компонента приложения рекомендуется создать отдельный пакет. Предположим, приложение должно сообщаться с веб-службой посредством XML и протокола HTTP для представления доменной модели клиента, причем информация о клиентах извлекается с помощью библиотечных классов. В этом случае рекомендуется создать два дополнительных пакета: для моделей и для запросов HTTP:

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

В этих пакетах будут находиться соответствующие классы Java. Тогда структура проекта Android будет выглядеть, как показано на рис. 3.27. Два дополнительных пакета вложены в пакет `com.dummies.android.helloandroid`, потому что их имена содержат имя данного пакета. Следовательно, для подчиненных компонентов можно создавать вложенные пакеты, чтобы структура проекта правильно отображала структуру компонентов приложения.

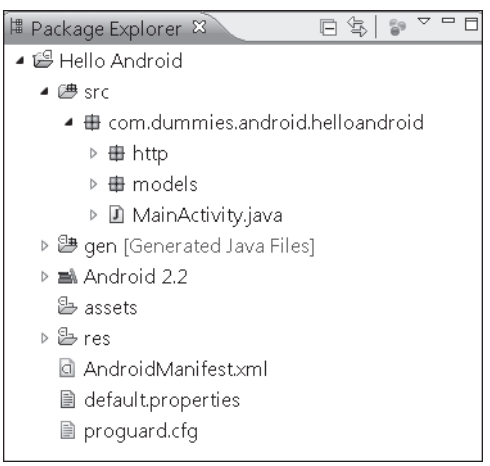


Рис. 3.27. Дополнительные пакеты находятся в пакете `com.dummies.android.helloandroid`.

Папка Android 2.2

Минутку! Мы пропустили папку gen! Не беспокойтесь, мы рассмотрим ее позже, вместе с папкой res. Сейчас остановимся подробнее на целевой версии Android. Собственно, это даже не папка как таковая (в файловой системе папки Android 2.2 нет), а, скорее, представление ресурсов Eclipse и Java надстройкой ADT.

Папка Android 2.2 содержит файл android.jar, версия которого определяется целевой версией Android, заданной в окне New Android Project при создании проекта. В файле android.jar находится заданная версия пакета разработки Android SDK. Развернув узел Android 2.2 и файл android.jar, можно увидеть инструменты и библиотеки, доступные для приложения (рис. 3.28). Все они были установлены во время инсталляции надстройки ADT в программу Eclipse.

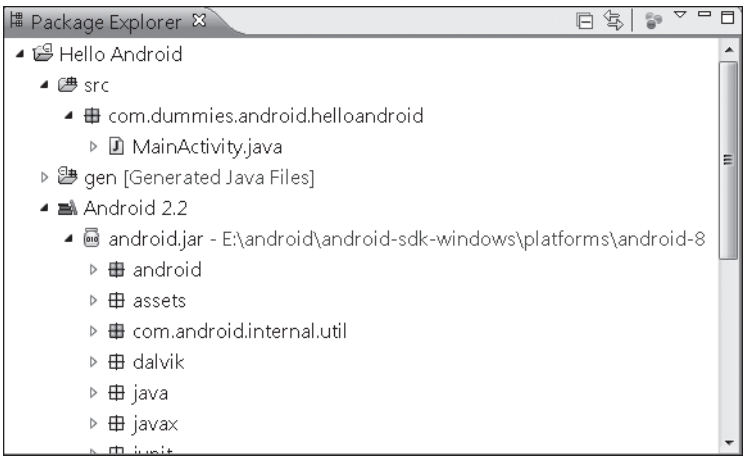


Рис. 3.28. Местонахождение и содержимое файла android.jar

Обратите внимание на то, что имя и местонахождение папки android вы задали произвольным образом во время установки надстройки ADT. Это говорит о том, что пакет Android SDK (как и программа Eclipse) может находится в любом месте файловой системы.

Папка assets

Сначала папка assets (активы) пустая. В нее можно помещать файлы, необходимые для приложения.

Актив — это файл данных любого типа и в любом формате, используемый в приложении. Часто файл актива содержит данные в проприетарном формате. Для обращения к активам, хранящимся в папке assets, удобно использовать класс AssetManager. Прочитав актив, приложение может прочитать хранящиеся в нем данные. Предположим, в приложении нужно использовать словарь специальных терминов (например, для реализации автозавершения). В таких случаях принято помещать словарь в папку assets (обычно это файл в формате XML или база данных SQLite).

В отличие от ресурсов, размещаемых в папке res и оснащаемых идентификаторами, активы могут содержать данные любого типа и не ассоциируются с

идентификаторами или обработчиками. Для каждого актива нужно создать обработчик вручную, например запрограммировать чтение потока битов и преобразование потока в аудио- или видеоформат.

Активам не присваиваются идентификаторы ID (в отличие от ресурсов). Для доступа к содержимому актива нужно (опять же, в отличие от ресурсов) вручную задать в коде обработку байтов, битов или потоков.

Папка `res`

Эта папка содержит ресурсы, используемые приложением. Рекомендуется всегда делать ресурсы внешними по отношению к коду приложения (но не по отношению к приложению). Классические примеры ресурсов — изображения и текстовые строки. Не размещайте строки в коде приложения. Вместо этого размещайте их в строковых ресурсах и обращайтесь к таким ресурсам из кода. Далее я покажу, как это делается. Ресурсы необходимо группировать в папке `res`.

Рекомендуется предоставлять альтернативные ресурсы для специфических конфигураций устройств путем группирования ресурсов в именованных вложенных папках. Во время выполнения операционная система Android выясняет, в какой конфигурации работает приложение, и выбирает соответствующий ресурс (или папку ресурса). Таким способом можно, например, отображать разные графические интерфейсы в зависимости от разрешения экрана или разные строки в зависимости от параметров локализации.

Сделав ресурс внешним, вы можете обратиться к нему в коде посредством идентификатора ресурса, сгенерированного автоматически надстройкой ADT в классе `R` (см. далее).

Необходимо помещать каждый тип ресурса в отдельную папку, вложенную в папку `res`. В табл. 3.1 приведены наиболее полезные типы ресурсов и соответствующие названия вложенных папок.

Таблица 3.1. Поддерживаемые типы ресурсов

Папка	Тип ресурса
`anim/`	Файлы XML, определяющие сеансы анимации
`color/`	Файлы XML, определяющие списки цветов
`drawable/`	Растровые файлы изображений (`.png`, `.jpg`, `.gif`) или файлы XML, скомпилированные в графические ресурсы
`drawable-hdpi/`	Графические ресурсы экранов с высокой разрешающей способностью. Суффикс `hdpi` указывает на высокое разрешение. Здесь хранятся те же ресурсы, что и в `drawable/`. Отличие заключается в том, что здесь ресурсы хранятся в скомпилированном виде
`drawable-ldpi/`	Графические ресурсы экранов с низкой разрешающей способностью. Суффикс `ldpi` указывает на низкое разрешение. Здесь хранятся те же ресурсы, что и в `drawable/`. Отличие заключается в том, что здесь ресурсы хранятся в скомпилированном виде

Папка	Тип ресурса
`drawable-mdpi/`	Графические ресурсы экранов со средней разрешающей способностью. Суффикс `mdpi` указывает на среднее разрешение. Здесь хранятся те же ресурсы, что и в `drawable/`. Отличие заключается в том, что здесь ресурсы хранятся в скомпилированном виде
`layout/`	Файлы XML, определяющие компоновку пользовательского интерфейса
`menu/`	Файлы XML, определяющие меню приложения
`raw/`	Произвольные файлы, хранящиеся в исходном формате. Файлы в этой папке не сжаты системой
`values/`	Файлы XML, содержащие простые значения, такие как текстовые строки, целые числа, цвета и т.п. Сравните — файл ресурса XML в папке `res` определяет один ресурс на основе имени файла, а файл в папке `values/` может определять много ресурсов разного назначения. Ниже приведен ряд соглашений об именовании ресурсов в этой папке: **`arrays.xml`** — массивы ресурсов (в одном файле хранится набор текстовых строк или целых чисел); **`colors.xml`** — ресурсы, определяющие цвета (доступны посредством класса `R.colors`); **`dimens.xml`** — ресурсы, определяющие размеры (например, `20px` означает 20 пикселей); доступны посредством класса `R.dimens`; **`strings.xml`** — строковые значения (доступны посредством класса `R.strings`); **`styles.xml`** — ресурсы, определяющие стили, аналогичные стилям CSS в формате HTML; в приложении можно определить разные стили, наследующие свойства друг друга (доступны посредством класса `R.styles`)

Никогда не размещайте файлы ресурсов непосредственно в папке `res`, потому что в этом случае компилятор сгенерирует сообщение об ошибке.

Ресурсы, которые сохраняются в папках, приведенных в табл. 3.1, называются *ресурсами, определенными по умолчанию*. Это означает, что они определяют установленные по умолчанию параметры структуры и компоновки приложения Android. Другие типы устройств Android могут потребовать иных типов ресурсов.

Используемый в Android механизм ресурсов очень мощный. О нем можно было бы написать не одну главу, однако в данной книге мы рассмотрим только базовые способы их использования, достаточные для того, чтобы вы могли создавать приложения Android. Ресурсы облегчают локализацию приложения, т.е. предоставляют способы адаптации приложения к разным языкам и странам. Кроме того, ресурсы облегчают адаптацию приложения к разным параметрам, предпочтениям и режимам мобильных устройств. Подробную информацию об использовании ресурсов можно найти в документации Android:

```
http://d.android.com/guide/topics/resources/providing-resources.html
```

Папки bin, libs и Referenced Libraries

В рассматриваемом примере (в приложении Hello Android) этих папок нет, но вы должны знать о них, чтобы при необходимости их использовать. Папка `libs` содержит закрытые библиотеки (т.е. библиотеки, доступные только для данного приложения). По умолчанию папка `libs` не создается. Если в проекте необходимо использовать закрытые библиотеки, нужно создать эту папку вручную. Для этого в окне Package Explorer (Обозреватель пакетов) щелкните правой кнопкой мыши на имени проекта и выберите в контекстном меню команду New⇨Folder (Создать⇨Папка). В появившемся диалоговом окне нужно задать имена родительской и создаваемой папок. В качестве родительской папки выберите проект Hello Android, введите имя создаваемой папки (т.е. `libs`) и щелкните на кнопке Finish (Готово).

В качестве закрытых библиотек обычно используются проприетарные библиотеки сторонних поставщиков, содержащие нужные функции. Например, jTwitter — это библиотека на Java для программного интерфейса твиттера. Чтобы применить библиотеку jTwitter в приложении Android, нужно поместить файл `jtwitter.jar` в папку `libs`. После этого щелкните правой кнопкой мыши на папке `libs` и выберите в контекстном меню команду Refresh (Обновить), чтобы библиотека `jtwitter.jar` появилась в окне обозревателя пакетов.

Кроме помещения библиотеки в папку `libs` ее нужно также добавить в маршрут сборки Java, т.е. задать маршрут классов, используемых при сборке проекта. Если проект зависит от других библиотек, программа Eclipse должна знать, где искать нужную библиотеку. Предположим, в папку `libs` добавлена библиотека `jtwitter.jar`. Тогда, чтобы добавить ее в маршрут сборки, щелкните правой кнопкой мыши на файле `jtwitter.jar` в окне обозревателя пакетов и выберите в контекстном меню команду Build Path⇨Add to Build Path (Маршрут сборки⇨Добавить в маршрут сборки).

Сразу после выбора этой команды автоматически создается папка `Referenced Libraries` (рис. 3.29). В файловой структуре компьютера этой папки нет. Она существует только в окне обозревателя пакетов, чтобы проинформировать вас о том, на какие библиотеки ссылается проект Eclipse.

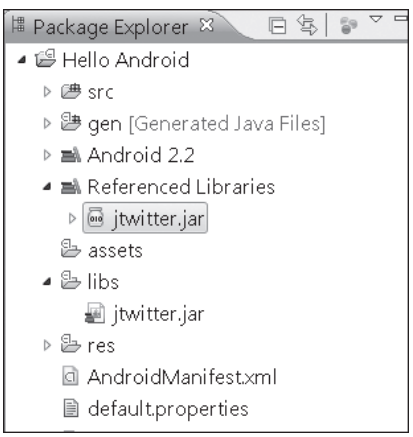


Рис. 3.29. Папка `Referenced Libraries`

Описание библиотеки jTwitter можно найти по такому адресу:

`http://www.winterwell.com/software/jtwitter.php`

В данной книге папка `libs` не используется. Однако если вы планируете стать профессиональным разработчиком, то будете использовать ее почти в каждом реальном приложении.

Папка gen

При создании приложения Android перед первой компиляцией папки `gen` не существует. Во время первой компиляции надстройка ADT генерирует папку `gen` и ее содержимое.

В первую очередь ADT создает файл `R.java` (подробнее о нем говорится далее). Выше рассматривалась папка `res`. Папка `gen` содержит сущности, сгенерированные на основе содержимого папки `res`. Поэтому для понимания назначения папки `gen` необходимо знать, что находится в папке `res`.

При написании кода Java для Android обязательно появляется необходимость сослаться на ресурсы, т.е. содержимое папки `res`. Это делается с помощью класса `R.java`. Файл `R.java` служит индексом всех ресурсов, определенных в папке `res`. Фактически этот класс — всего лишь сокращенный инструмент ссылки на ресурсы, включенные в проект. Класс `R` особенно полезен благодаря средству автозавершения кода, встроенному в Eclipse. Благодаря классу `R` можно быстро идентифицировать нужный ресурс в окне подсказки.

В проекте Hello Android разверните папку `gen` и пакет, находящийся в папке `gen`. Дважды щелкните на файле `R.java`, чтобы открыть его на правой панели. В нем находятся вложенные классы Java. Эти вложенные классы имеют те же имена, что и соответствующие папки, вложенные в папку `res`. В каждом вложенном классе объявлены члены с теми же именами, что и соответствующие ресурсы в папке `res` (исключая расширения файлов). Ниже приведен автоматически сгенерированный код файла `R.java`.

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found. It
* should not be modified by hand.
АВТОМАТИЧЕСКИ СГЕНЕРИРОВАННЫЙ ФАЙЛ. ВРУЧНУЮ НЕ ИЗМЕНЯТЬ.

Этот класс был автоматически сгенерирован с помощью
инструмента aapt на основе найденных ресурсов данных.
Изменять содержимое класса вручную нельзя.
*/
package com.dummies.android.helloandroid;

public final class R {
  public static final class attr {
  }
  public static final class drawable {
```

```
    public static final int icon=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;
  }
}
```

Что это за странные числа, начинающиеся с `0x`? Рад сообщить, что программисту о них можно не беспокоиться. Надстройка ADT генерирует этот код для вас, чтобы вы могли не думать о том, что происходит "за кулисами". Когда вы добавляете ресурсы и проект повторно собирается, надстройка ADT автоматически регенерирует файл `R.java`. Обновленный файл содержит члены, ссылающиеся на добавленные вами ресурсы.

Никогда не редактируйте файл `R.java` вручную. Если вы сделаете это, приложение может не скомпилироваться. Если вы изменили файл `R.java` случайно и не можете отменить изменения, удалите папку `gen` и соберите проект заново. Тогда надстройка ADT регенерирует файл `R.java`.

Файл манифеста приложения

Отслеживать все, что вам нужно, можно с помощью списков, не так ли? Именно для этого в Android существует файл манифеста. Он отслеживает все, что необходимо приложению, что оно запрашивает и что оно должно использовать во время выполнения.

Файл манифеста Android хранится в корневой папке проекта и называется `AndroidManifest.xml`. Каждое приложение должно иметь файл манифеста Android в своей корневой папке.

Файл манифеста предоставляет операционной системе Android всю информацию, необходимую для выполнения кода приложения. В частности, файл манифеста должен предоставлять следующую информацию.

- ✓ Имя пакета Java, в котором размещено приложение. Имя должно быть идентификатором, уникальным в операционной системе (т.е. в физическом устройстве) и, желательно, в Android Market.
- ✓ Списки компонентов приложения, таких как деятельности и фоновые службы.
- ✓ Объявление разрешений, запрашиваемых приложением при выполнении.
- ✓ Минимальная версия Android API, необходимая для работы приложения.

В файле манифеста обязательно должна быть объявлена версия приложения. Важно в самом начале работы над приложением определить стратегию управления версиями, включая нумерацию версий будущих релизов приложения. Каждое приложение должно иметь пользовательский номер версии (version name) и код версии (version code). Эти два значения рассматриваются в следующих разделах.

Код версии

Код версии (другое название — *внутренний номер версии*) представляет собой целое число, обозначающее версию кода приложения относительно других версий. Значение кода версии данного приложения используется другими приложениями для определения совместимости с данным приложением. Кроме того, на сайте Android Market коды версий используются для внутренней идентификации приложений и управления обновлениями. Не путайте код версии приложения с кодом версии Android, используемой данным приложением (см. выше).

Коду версии можно присвоить любое целочисленное значение. Нужно лишь соблюдать такое правило: код версии каждого следующего релиза должен быть больше кода версии предыдущего релиза (и, соответственно, всех предыдущих). Операционная система Android не вынуждает соблюдать это правило; ответственность за его соблюдение лежит на вас.

Обычно для первого релиза коду версии присваивается значение 1. Затем для каждого следующего релиза код версии увеличивается на единицу независимо от того, изменился ли старший (major) или младший (minor) пользовательский номер версии. Это означает, что внешне код версии не похож на номер версии, видимый пользователем. Обычно код версии для пользователя не отображается.

Если вы обновите код приложения и поставите новый релиз, не увеличив код версии, то разные кодовые базы приложения будут иметь один и тот же код версии. Предположим, первый релиз вашего приложения имеет код версии 1. Пользователь устанавливает ваше приложение, загрузив его с Android Market, находит в приложении ошибку и сообщает вам о ней. Вы исправляете ошибку, компилируете новую кодовую базу и поставляете релиз на Android Market, не обновив код версии в файле манифеста. Сайт Android Market не знает, что изменился код приложения, потому что он проверяет только код версии в файле манифеста приложения. Соответственно, сайт никак не реагирует на изменение версии приложения. Если же вы изменили код версии, присвоив ему значение 2, сайт Android Market обнаружит это и предложит пользователям версии 1 загрузить и установить версию 2. Если не изменить код версии, пользователи никогда не узнают, что появилась новая версия приложения, в которой исправлена ошибка.

Пользовательский номер версии

Пользовательский номер версии (version name) — это строковое значение, обозначающее версию кода приложения и видимое пользователю. Номер версии похож на число, но на самом деле это строка, подчиняющаяся следующему синтаксису:

```
<старший_номер>.<младший_номер>.<номер_сборки>
```

Номер сборки часто опускают, тогда, например, номер версии 2.1.4 выглядит как 2.1.

Операционная система Android никак не реагирует на пользовательский номер версии, она лишь отображает его для пользователя.

Выше приведен лишь один из возможных (хотя и наиболее распространенный) формат номера версии. Существуют и другие форматы. Например, в приложении Foursquare используется схема нумерации версий с привязкой к датам. Версия `2010-06-08` означает, что данный релиз был выпущен 8 июня 2010 года. В принципе, можете нумеровать версии, как вам захочется, нужно лишь, чтобы ваша нумерация была понятна другим людям, включая пользователей.

Разрешения

Предположим, приложению нужен доступ к Интернету для получения некоторых данных. По умолчанию операционная система Android ограничивает доступ к Интернету. Чтобы получить его, нужно запросить разрешение на эту операцию.

В файле манифеста приложения нужно указать, какие разрешения нужны приложению для правильной работы. В табл. 3.2 приведены некоторые из часто запрашиваемых разрешений.

Таблица 3.2. Примеры разрешений

Разрешение	Описание
Internet	
Write External Storage	
Camera	
Access Fine Location	
Read Phone State	

Файл `default.properties`

Этот файл используется совместно надстройкой ADT и программой Eclipse. Он содержит свойства проекта (например, такие, как целевая платформа компиляции) и является частью проекта.

Файл `default.properties` нельзя редактировать вручную. Для изменения его содержимого используется окно редактора свойств, встроенного в Eclipse (рис. 3.30). Чтобы открыть его, щелкните правой кнопкой мыши на имени проекта в окне Package Explorer (Обозреватель пакетов) и выберите в контекстном меню команду Properties (Свойства).

На левой панели редактора приведены категории свойств. Чтобы изменить значение какого-либо свойства, нужно выбрать его категорию на левой панели. Тогда свойство появится на правой панели, и можно будет отредактировать его. Например, выбрав на левой панели категорию Android, можно изменить маршрут пакета Android SDK.

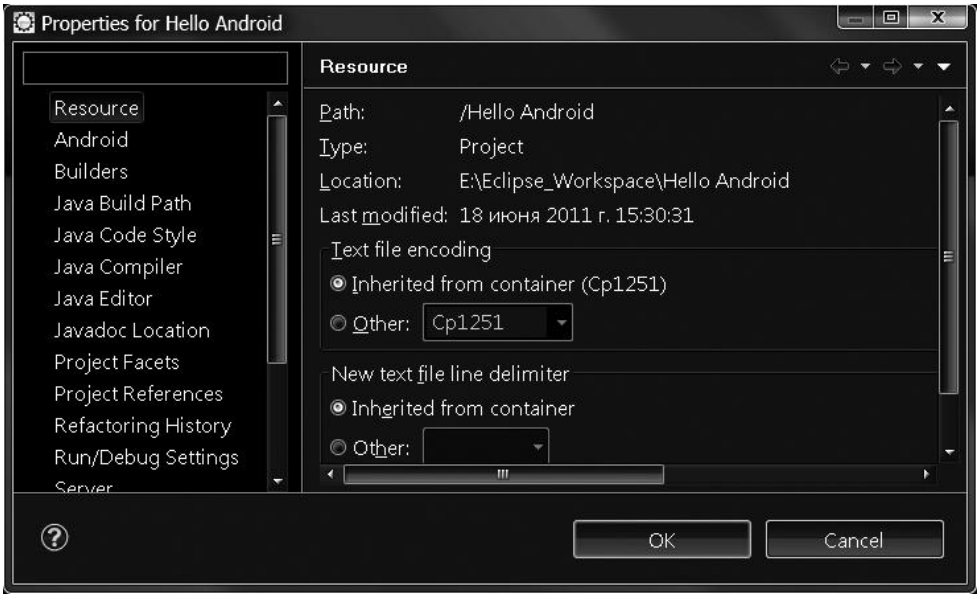


Рис. 3.30. Окно редактора свойств проекта

Глава 4

Разработка пользовательского интерфейса

В этой главе...

- Создание проекта Silent Mode Toggle
- Компоновка приложения
- Разработка пользовательского интерфейса
- Добавление изображения в приложение
- Создание значка запуска приложения
- Добавление кнопки
- Приложение в режиме конструктора

Итак, вы уже знаете, что такое Android, и способны создать простое приложение. Значит, мы можем перейти к самому интересному — к созданию реального приложения, которое можно использовать в повседневной жизни или опубликовать на сайте Android Market.

Приложение, которое вы создадите в этой главе, позволяет переключить режим телефонного звонка в результате одного нажатия кнопки. Это очень полезное приложение, и его захочет иметь в своем телефоне каждый пользователь. Следовательно, если бы вы были единственным читателем данной книги, то смогли бы заработать на продаже этого приложения кучу денег. Впрочем, еще не все потеряно. Никто не запрещает вам придумать полезную модификацию данного приложения и выставить его на продажу под своим именем.

Представьте себе, что вы находитесь на рабочем месте и администрация фирмы пригласила вас на важное совещание. Естественно, перед началом совещания вы переключаете звонок в бесшумный режим (или виброрежим). Ведь не хотите же вы, чтобы в самый напряженный момент совещания ваш телефон неожиданно зазвонил и на вас обратились неодобрительные взгляды всех присутствующих. Когда совещание закончится, вы переключите телефон в режим громкого звонка. Однако вы не хотите, чтобы звонок был самым громким и все окружающие вздрагивали от неожиданности. Вы предпочитаете определенный уровень громкости. Следовательно, при каждом включении звонка вам нужно заново настроить его громкость.

Учитывая, что при каждом переключении режима звонка нужно открыть меню и найти нужные подменю, а также то, что совещания могут созываться часто, можно предположить, что ваша жизнь весьма хлопотливая. Чтобы хотя бы немного облегчить ее, давайте сделаем так, чтобы режим звонка переключался после одного нажатия кнопки. Это будет делать наше приложение. Причем при включении звонка не нужно будет заново настраивать его громкость. Она остается на том же уровне, на котором была в момент отключения звонка.

Создание проекта Silent Mode Toggle

Сейчас мы создадим приложение Silent Mode Toggle (Переключение режима сигнала). Вы уже знаете, как создать пустой проект Android в рабочей среде Eclipse, поэтому в данном вопросе я не буду "вести вас за руку". Если забыли, как это делается, прочитайте еще раз главу 3.

Перед созданием нового проекта закройте все файлы, открытые в рабочей среде Eclipse. Для этого на правой панели Eclipse по очереди щелкните на крестике на корешке каждого файла. Можете также выбрать в главном меню команду File⇨Close All (Файл⇨Закрыть все).

Закрыв все файлы, закройте также текущий проект Hello Android, с которым вы работали, читая предыдущую главу. Для этого в окне Package Explorer (Обозреватель папок) рабочей среды Eclipse щелкните правой кнопкой мыши на проекте Hello Android и выберите в контекстном меню команду Close Project (Закрыть проект). Этим вы сообщаете программе Eclipse, что некоторое время не будете работать с данным проектом. Программа Eclipse освободит ресурсы, используемые для отслеживания состояния проекта, а следовательно, скорость выполнения всех операций в Eclipse увеличится.

Теперь все готово для создания приложения Silent Mode Toggle. Сначала создайте проект. Для этого выберите команду File⇨New Project (Файл⇨Создать проект). Выберите в списке тип проекта Android Project и щелкните на кнопке Next (Далее). Установите свойства проекта согласно табл. 4.1.

Таблица 4.1. Свойства проекта Silent Mode Toggle

Параметр	Значение
Application Name (Имя приложения)	`Silent Mode Toggle`
Project Name (Имя проекта)	`Silent Mode Toggle`
Contents (Содержимое)	Оставьте значения, установленные по умолчанию
Build Target (Целевая платформа)	Android 2.2
Package Name (Имя пакета)	`com.dummies.android.silentmodetoggle`
Create Activity (Создать деятельность)	`MainActivity`
Min SDK Version (Минимальная версия SDK)	`8`

Щелкните на кнопке Finish (Готово). Приложение Silent Mode Toggle должно отображаться в окне обозревателя пакетов (рис. 4.1).

Иногда при создании проекта рабочая среда Eclipse возвращает сообщение об ошибке, подобное следующему: `The project cannot be built until build path errors are resolved` (Проект нельзя собрать, пока не устранены ошибки маршрута сборки). В этом случае щелкните правой кнопкой мыши на имени проекта и выберите в контекстном меню команду Android Tools⇨Fix Project Properties (Инструменты Android⇨Исправить свойства проекта). Программа Eclipse автоматически синхронизирует маршруты проекта и рабочего пространства.

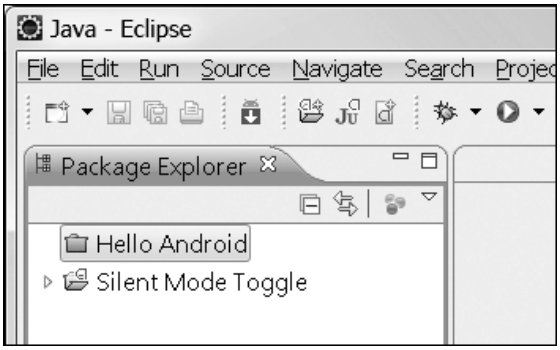


Рис. 4.1. Приложение Silent Mode Toggle в рабочей среде Eclipse

Выбирая параметр Build Target, вы определяете, какие инструменты будут использоваться при создании приложения. Выбирая Min SDK Version, вы сообщаете пользователям, какая минимальная версия Android может "переварить" ваше приложение. Вам выгоднее задать как можно более низкую минимальную версию, потому что, чем ниже версия, тем шире потенциальная аудитория вашего приложения. Но как узнать, какова минимальная версия для данного приложения? Единственный способ узнать это состоит в том, чтобы попробовать запустить и протестировать приложение в устаревшей среде.

Компоновка приложения

Теперь, когда проект Silent Mode Toggle определен в рабочей среде Eclipse, можно приступить к разработке графического пользовательского интерфейса приложения. Пользовательский интерфейс — это та часть приложения, посредством которой пользователь взаимодействует с устройством. Фактически он является "лицом приложения". Его внешний вид определяет впечатление пользователя от приложения, а компоновка интерфейса определяет удобство работы с приложением.

Приложение Silent Mode Toggle содержит единственную кнопку, расположенную в центре экрана и переключающую режим звонка с бесшумного на громкий или наоборот. Над кнопкой находится изображение, которое визуально извещает пользователя о том, в каком режиме сейчас работает телефон: бесшумном или громком. Лучше один раз увидеть, чем сто раз услышать, поэтому давайте посмотрим на рис. 4.2 и 4.3, чтобы узнать, как должно выглядеть приложение в первом и втором режимах. Считайте эти два рисунка своего рода "техническим заданием", сообщающим, как должно выглядеть создаваемое вами приложение. В то же время рассматривайте это как догму. Поэкспериментируйте с размерами и содержимым изображения в этих режимах, размерами, цветом и шрифтом надписи на кнопке. Возможно, вам удастся создать эстетически более привлекательный интерфейс. В частности, попытайтесь создать небольшой зазор между изображением и кнопкой.

Рис. 4.2. Приложение в бесшумном режиме

Рис. 4.3. Приложение в громком режиме

Использование файла компоновки XML

В рабочем пространстве Eclipse все файлы компоновки приложения хранятся в папке `res/layouts` проекта Android. Когда вы несколько минут назад создавали пустой проект Silent Mode Toggle, надстройка ADT создала для вас файл `main.xml` и поместила его в папку `res/layouts`. Это файл компоновки пользовательского интерфейса, устанавливаемого по умолчанию. Надстройка ADT автоматически создает его для каждого нового приложения. Откройте файл `main.xml`, дважды щелкнув на его имени на левой панели редактора Eclipse (рис. 4.4). На правой панели будет отображено содержимое файла.

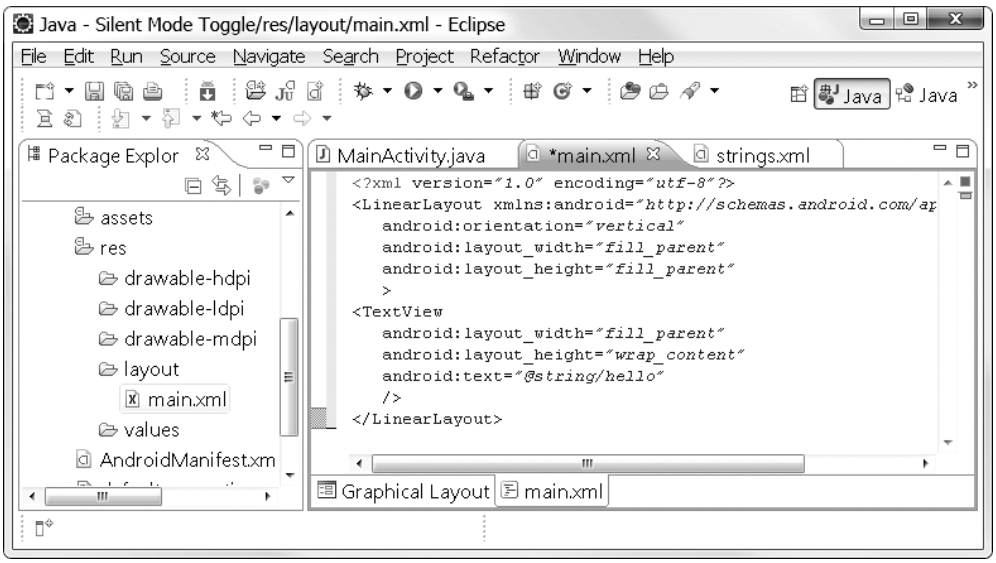


Рис. 4.4. Текст файла `main.xml`

Простая разметка, представленная на рис. 4.4, определяет такой же простой пользовательский интерфейс, отображающий текстовое поле в середине экрана. Ниже приведен код файла `main.xml`.

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

Содержимое файла XML однозначно определяет внешний вид пользовательского интерфейса. Ввиду исключительной важности этого файла для всей вашей дальнейшей карьеры разработчика подробно рассмотрим каждый элемент разметки.

Объявление документа XML

Первый элемент — объявление документа XML — сообщает текстовому редактору Eclipse и операционной системе Android, что это файл XML, а также указывает версию XML и кодировку файла.

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

Тип компоновки

Следующий элемент (`LinearLayout`) определяет линейный тип компоновки пользовательского интерфейса. Более подробно элемент `LinearLayout` рассматривается далее, а сейчас отметим лишь, что он служит контейнером для визуальных элементов графического интерфейса, отображаемых на экране и называемых "представлениями". Атрибуты элемента `LinearLayout` определяют способ и параметры компоновки представлений. Ниже приведен код открывающего дескриптора `LinearLayout`. Закрывающий дескриптор `</LinearLayout>` здесь не приведен, потому что элемент `LinearLayout` является контейнером, и, следовательно, закрывающий дескриптор расположен после всех элементов представлений, добавленных в контейнер.

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

Представления

На платформе Android *представление* (view) является базовым элементом пользовательского интерфейса. Представления — своего рода строительные блоки, из которых состоит графический интерфейс пользователя. Каждое представление — это, с одной стороны, экземпляр некоторого класса (например, класса `TextView`), а с другой стороны, элемент интерфейса, видимый на экране. В разметке XML дескриптор представления определяет визуальные свойства видимого элемента пользовательского интерфейса. Ниже приведена разметка представления `TextView`, которое отображает на экране текстовую строку, хранящуюся в ресурсе `@string/hello` (ресурсы рассматривались в главе 3).

```
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello"
   />
```

Представление занимает на экране прямоугольную область и отвечает за прорисовку и обработку событий элемента интерфейса. Все визуальные элементы, видимые на экране мобильного устройства, являются представлениями. Класс `View` является базовым (родительским) для всех стандартных визуальных элементов Android, следовательно, они наследуют поля и методы класса `View`.

В конце файла компоновки находится приведенный ниже закрывающий дескриптор элемента `LinearLayout`, который сообщает о том, что больше элементов нет.

```
</LinearLayout>
```

Типы компоновок

При создании пользовательского интерфейса нужно как-то размещать его визуальные элементы на экране. Способ их размещения называется *компоновкой*. Линейная компоновка — это размещение их один за другим (по вертикали или горизонтали). Иногда нужно размещать элементы в ячейках таблицы, а иногда — задавать координаты каждого элемента (такой способ называется *точным позиционированием*). К счастью, разработчики Android создали для нас много инструментов, позволяющих применять разные типы компоновки. Каждому типу компоновки соответствует определенный класс Java, каждому из которых в свою очередь соответствует определенный дескриптор XML в файле `main.xml`. В табл. 4.2 приведены наиболее популярные типы компоновки, доступные в Android SDK.

Таблица 4.2. Типы компоновок, доступные в Android SDK

Имя класса	Описание
`LinearLayout`	Дочерние элементы контейнера размещаются в один ряд
`RelativeLayout`	Позиции дочерних элементов определяются по отношению друг к другу или родительскому элементу
`FrameLayout`	Этот контейнер блокирует часть экрана для отображения единственного элемента. В элемент `FrameLayout` можно добавить много дочерних элементов, но все они будут пристыкованы в левому верхнему углу контейнера. Дочерние элементы прорисовываются в порядке их размещения в разметке, следовательно, более ранние элементы закрываются более поздними
`TableLayout`	Дочерние элементы размещаются по строкам и столбцам таблицы

Есть и другие классы компоновки, например `TabHost`, который создает вкладки, или `SlidingDrawer`, который обеспечивает сокрытие или отображение представления, когда пользователь проводит по нему пальцем (это похоже на движение, которым человек смахивает крошку со стола). Многие классы компоновки применяются только в специальных ситуациях. В реальной жизни чаще всего используются компоновки, приведенные в табл. 4.2.

Визуальная среда разработки

У меня для вас две новости: хорошая и плохая. Хорошая состоит в том, что в Eclipse есть визуальная среда разработки (окно конструктора). Плохая новость — средства визуальной разработки довольно ограниченные, поэтому создать мощный интерфейс, только щелкая мышью, вам не удастся. Придется время от времени писать коды. Впрочем, потом вы поймете, что это не так уж плохо, потому что только в разметке XML можно ясно увидеть, что куда вложено.

Открытие окна конструктора

На рис. 4.5 показан код файла `main.xml` в окне редактора кода. Чтобы открыть этот же файл в окне конструктора, щелкните на вкладке Graphical Layout (Графическая компоновка) в нижней части экрана.

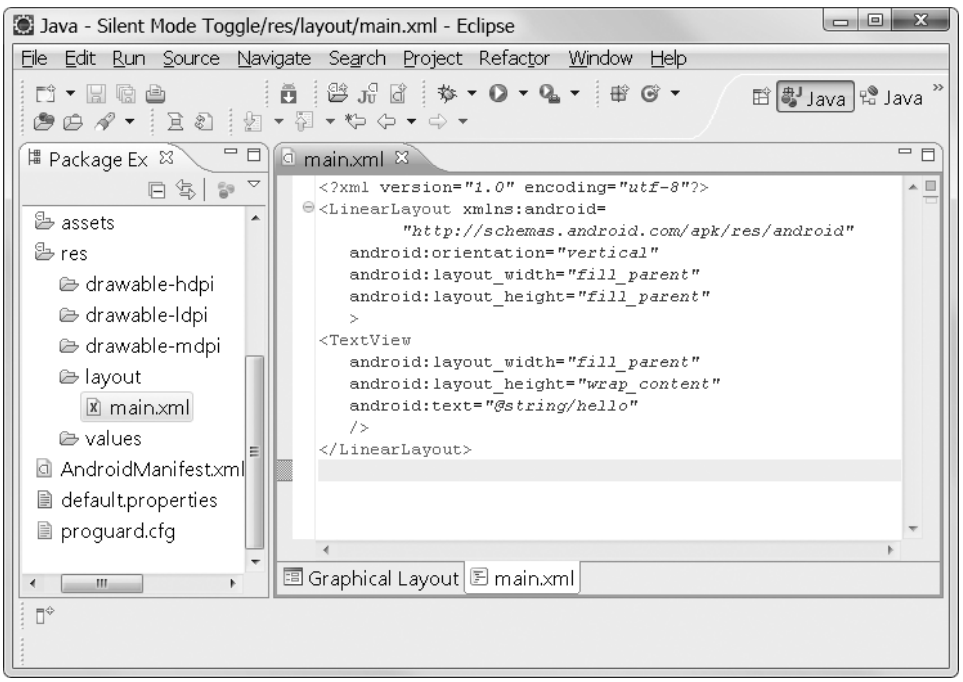


Рис. 4.5. Файл `main.xml` в режиме редактора кода

После щелчка на вкладке Graphical Layout этот же файл будет показан в окне конструктора (рис. 4.6).

В режиме конструктора можно перетаскивать представления и контейнеры с панели инструментов, расположенной слева, в окно приложения, расположенное справа. В данный момент в окне приложения есть единственный элемент — представление `TextView`, отображающее фразу `Привет, файл main.xml`. Его же можно увидеть в окне редактора.

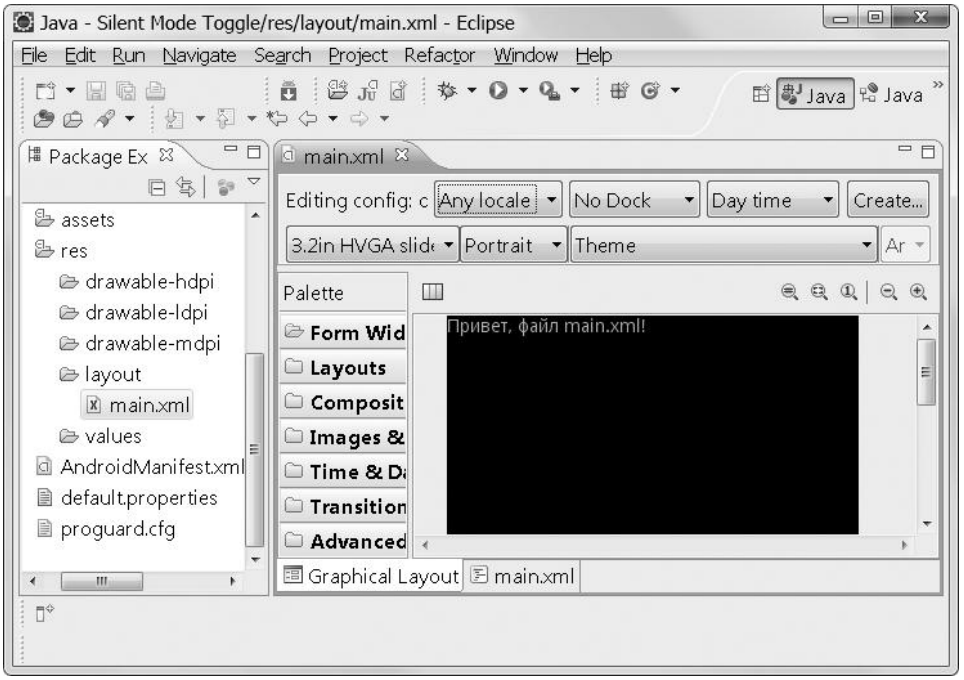


Рис. 4.6. Файл `main.xml` в режиме конструктора

Редактирование свойств представлений

В режиме конструктора можно просматривать и редактировать свойства любого представления. Для этого нужно щелкнуть на нем, чтобы выделить его. Свойства выделенного представления отображаются на панели Properties (Свойства). Если панель Properties отсутствует, выполните следующие операции, чтобы открыть ее.

1. **Щелкните на представлении, чтобы выделить его.**
2. **Щелкните на представлении правой кнопкой мыши и выберите в контекстном меню команду Show In⇨Properties (Показать в⇨Свойства).**

   На экране будет отображено окно свойств (рис. 4.7). В левом столбце приведены имена свойств, а в правом — их значения. При щелчке в поле значения происходит одно из двух: либо в поле устанавливается курсор ввода (тогда можно ввести нужное значение с клавиатуры), либо в правом конце поля появляется стрелочка, позволяющая выбрать нужное значение в раскрывающемся списке.

Окно Properties часто бывает полезным в качестве своеобразного справочника по свойствам представлений. Ни один разработчик не помнит наизусть весь список свойств. Найти нужное свойство в окне Properties легче, чем в документации по Android, потому что окно Properties всегда “под рукой”. Открыть окно Properties можно, выбрав команду Windows⇨Show View⇨Other⇨General⇨Properties (Окна⇨Показать окна⇨Другие⇨Общие⇨Свойства).

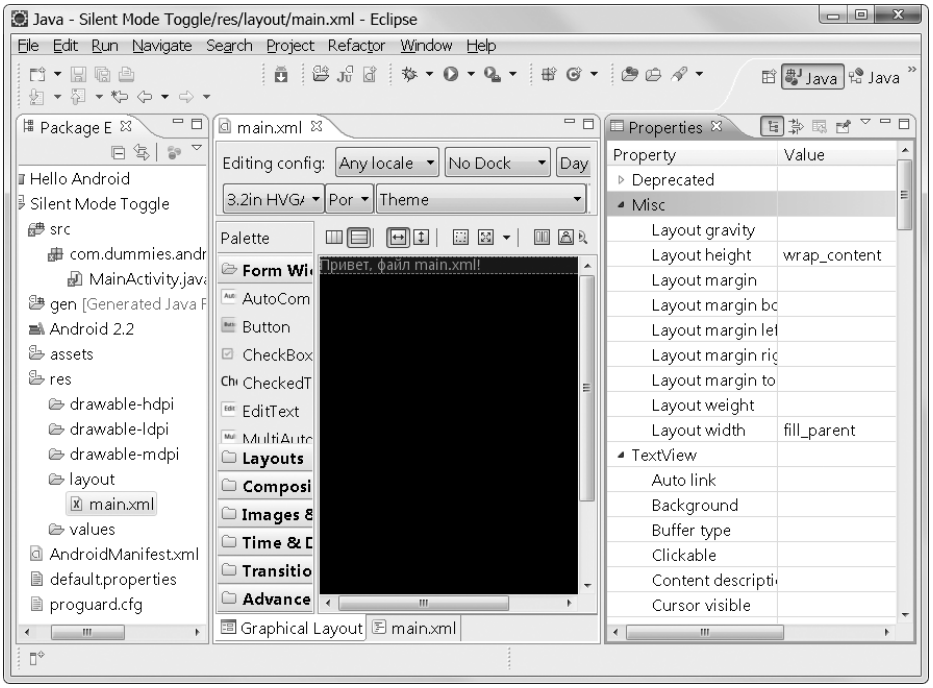


Рис. 4.7. В окне свойств (справа) отображаются свойства выделенного представления

Список доступных свойств представления может изменяться в зависимости от типа компоновки родительского контейнера. Например, в контейнере `LinearLayout` список свойств представления `TextView` не такой же, как в контейнере `RelativeLayout`.

Режим конструктора очень полезен для решения статических задач, когда визуальное содержимое контейнера закреплено в одном месте и размеры представлений не изменяются. Однако, если представления должны отображаться по-разному в зависимости от результатов вычисления или данных, вводимых пользователем, режим конструктора не очень полезен. Конечно, в окне конструктора можно определить начальные позиции динамических представлений, но все дальнейшие изменения и перемещения придется задавать в коде. То же относится и к внутреннему содержимому представлений. Например, представление `TextView` можно разместить на экране в режиме конструктора. Можно даже определить в разметке файла `main.xml` начальный текст, но изменять отображаемый текст можно только в коде Java.

Разработка пользовательского интерфейса

В первую очередь отобразите на экране код файла `main.xml`, содержащий разметку XML. Для этого щелкните на вкладке `main.xml`, расположенной в нижней части экрана справа от Graphical Layout (Графическая компоновка), как показано на рис. 4.7. Удалите из разметки элемент `TextView`. Разметка должна выглядеть следующим образом.

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

Атрибуты дескриптора компоновки

В файле XML используется дескриптор компоновки `LinearLayout`, который неявно определяет соответствующие представление и класс `LinearLayout`. Свойства представления определяются атрибутами дескриптора XML. В табл. 4.3 приведено описание атрибутов, используемых в данном примере.

Таблица 4.3. Атрибуты компоновки

Атрибут	Описание
`xmlns:android="..."`	Определение пространства имен XML, используемого для ссылки на компоненты Android SDK
`android:orientation="vertical"`	Этот атрибут информирует контейнер о том, что представления должны размещаться в нем одно за другим по вертикали
`android:layout_width="fill_parent"`	Этот атрибут означает, что по горизонтали данный контейнер должен заполнить все доступное пространство
`android:layout_height="fill_parent"`	Этот атрибут означает, что по вертикали данный контейнер должен заполнить все доступное пространство

Итак, тип компоновки и параметры контейнера `LinearLayout` определены. Контейнер заполняет весь экран по горизонтали и вертикали. Осталось поместить в него нужные представления.

Размещение представлений в контейнере

Как уже отмечалось, представления — это своего рода строительные блоки, из которых состоит графический интерфейс пользователя. Создавая визуальный элемент графического интерфейса, вы создаете представление. При работе с представлениями в Java их нужно приводить к соответствующим типам.

Установка параметров `layout_width` и `layout_height`

Прежде чем представление появится на экране, нужно сконфигурировать его свойства, чтобы операционная система Android знала, как нарисовать его. Свойства представления задаются значениями атрибутов, приведенных в дескрипторе представления. Атрибуты `layout_width` (ширина компоновки) и `layout_height` (высота компоновки) являются обязательными. В Android SDK они относятся к классу `LayoutParams`.

Атрибут `layout_width` определяет ширину представления, а `layout_height` — его высоту.

Значения fill_parent и wrap_contents

Атрибуты layout_width и layout_height могут принимать значения в пикселях или предопределенные строковые значения. Два наиболее распространенных предопределенных строковых значения — fill_parent (заполнить родительский) и wrap_content (сворачивать содержимое).

Значение fill_parent информирует операционную систему Android о том, что представление должно заполнить собой все пространство, доступное в родительской компоновке. При значении wrap_content Android должна выделять для представления ровно столько пространства, сколько нужно для отображения его содержимого. При этом, когда объем содержимого увеличивается (например, удлиняется строка, отображаемая элементом TextView), расширяется также пространство, занимаемое представлением на экране. Значение wrap_content аналогично свойству Autosize (Автоматический размер) в формах Windows.

Если используется статическая компоновка, два указанных атрибута можно установить в файле XML. Если же компоновка динамическая (т.е. должна изменяться во время работы пользователя с устройством), параметры размещения должны устанавливаться в коде Java, где они представлены не как значения атрибутов, а как поля классов, причем имена полей и атрибутов могут не совпадать. В любом случае вы должны обязательно присвоить значения параметрам компоновки. Динамические представления в данной книге не рассматриваются. Примеры динамических представлений можно найти в образцах кодов, поставляемых в пакете Android SDK.

Если забудете установить значения атрибутов layout_width и layout_height, приложение потерпит крах во время выполнения при попытке вывода представления. К счастью, это легко обнаруживается при тестировании приложения.

В версии Android 2.2 значение fill_parent было переименовано в match_parent. Однако для обратной совместимости значение fill_parent продолжает поддерживаться, поэтому я использую его в данной книге. Но если вы планируете разрабатывать приложения для версий 2.2 и выше, то вам лучше использовать значение match_parent.

Добавление изображения в приложение

Приложение выглядело бы довольно скучным, если бы могло отображать на экране только текст, поэтому нужно добавить изображения. Как это делается, я расскажу в следующих разделах.

Размещение изображения на экране

В проекте Silent Toggle Mode мы добавим на экран изображение телефонной трубки (см. рис. 4.2 и 4.3), которое будет информировать пользователя о текущем режиме звонка (бесшумный или громкий). Поскольку есть два режима, нужны два изображения. Чтобы добавить изображения в приложение, нужно, естественно, иметь файлы изображений. Можете загрузить их с сайта данной книги или применить собственные изображения, какие вам больше нравятся.

Зачем нужны папки для разных разрешений экрана

Операционная система Android поддерживает разные размеры и разрешения экрана. Графические ресурсы можно размещать в трех разных папках (`drawable-ldpi`, `drawable-mdpi` и `drawable-hdpi`), соответствующих низкому, среднему и высокому разрешениям экрана. Предположим, одно и то же изображение используется при любых разрешениях экрана. К чему это приведет? К тому, что на экранах с низким разрешением изображение будет либо обрезано, либо сжато. И то, и другое плохо. А при высоком разрешении экрана изображение будет выглядеть крупнозернистым, что тоже плохо. Для устранения этих проблем создайте несколько версий изображения для разных разрешений экрана. Дополнительную информацию о методиках адаптации приложения Android к разным разрешениям экрана можно найти по следующему адресу:

`http://developer.android.com/guide/practices/screens-support.html`

Чтобы добавить изображения в проект, нужно перетащить их в соответствующие папки и создать ссылки на них в проекте. Для этого выполните следующие операции.

1. **Откройте рядом на экране два окна — проводника Windows и Eclipse. Перетащите изображение телефонной трубки (файл `phone_silent.png`) с правой панели проводника в папку `res/drawable-mdpi` на левой панели Eclipse (рис. 4.8).**

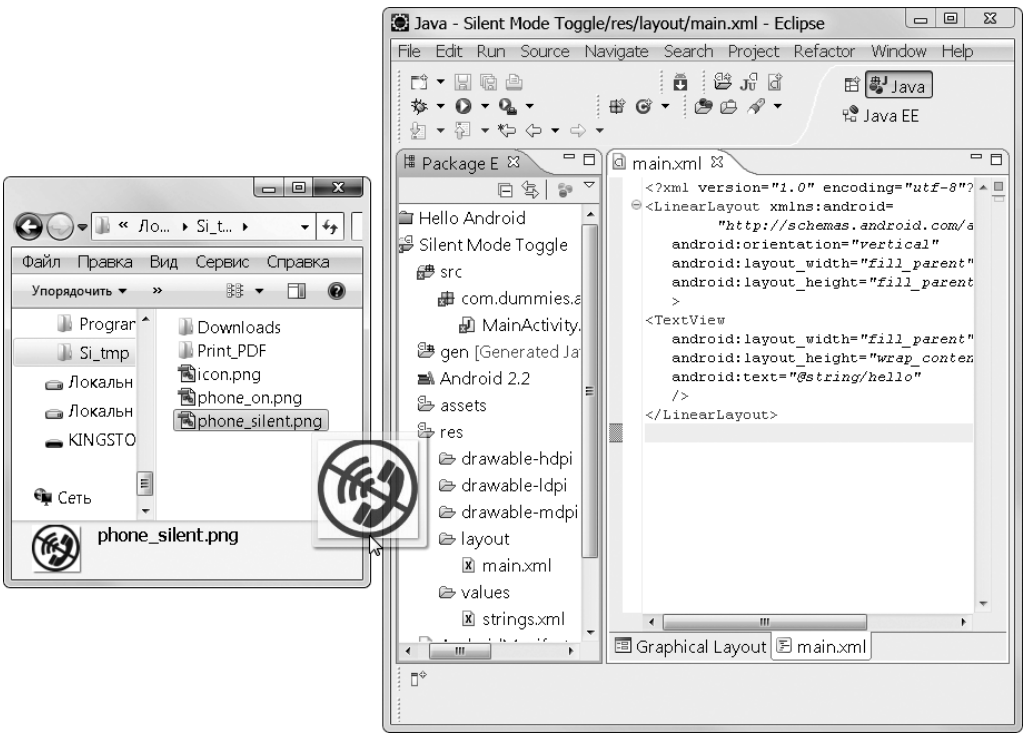


Рис. 4.8. Перетаскивание файла изображения

2. **Таким же образом перетащите файл `phone_on.png`.**

   В папке `drawable-mdpi` должно быть два файла изображений:

   - `phone_on.png`, обозначающий режим громкого звонка;
   - `phone_silent.png`, обозначающий бесшумный режим.

   Если файлы изображений имеют другие имена, можете переименовать их непосредственно в папке `drawable-mdpi`. Окно обозревателя пакетов сейчас должно выглядеть, как показано на рис. 4.9. Файл `icon.png` — это значок приложения Android, добавленный надстройкой ADT автоматически.

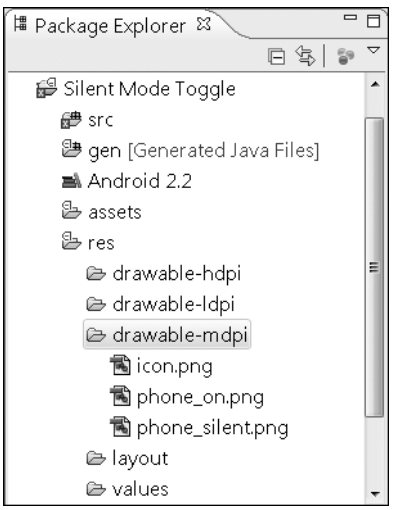


Рис. 4.9. В проект Silent Toggle Mode добавлены изображения

При перетаскивании файла изображения на панель Eclipse надстройка ADT обнаруживает, что структура проекта изменилась, и, если в меню Eclipse установлен флажок Project⇨Build Automatically (Проект⇨Автоматическая сборка), снова собирает проект. В частности, регенерируется папка `gen`, содержащая файл `R.java`. Теперь в файле `R.java` есть ссылки на два новых изображения. Ссылки на изображения можно использовать для обращения к изображениям в коде Java или XML. В следующем разделе мы объявим изображения в разметке XML.

Добавление изображения в разметку XML

Сейчас изображения `phone_on.png` и `phone_silent.png` есть в проекте и доступны для приложения, но никак не используются приложением. Чтобы приложение вывело изображение на экран, нужно создать элемент интерфейса `ImageView`, рисующий изображение на экране, и добавить в него ссылку на изображение. Для этого перепишите файл `main.xml` следующим образом.

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

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<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>
```

Вы добавили представление `ImageView` в контейнер `LinearLayout`. Класс `ImageView` прорисовывает изображение на экране мобильного устройства.

Установка свойств изображения

В классе `ImageView` определено также несколько дополнительных параметров, с которыми вы пока что не встречались.

- ✓ **`android:id="@+id/phone_icon"`.** Атрибут `id` определяет уникальный идентификатор представления в операционной системе Android. При выборе идентификатора рекомендуется придерживаться соглашений об именовании, приведенных по такому адресу:

  `http//developer.android.com/guide/topics/ui/declaring-layout.html`

- ✓ **`android:layout_gravity`.** Это свойство определяет, как должно быть размещено представление по отношению к родительскому контейнеру по обеим осям — горизонтальной и вертикальной. В рассматриваемом примере данному атрибуту присвоена константа `center_horizontal`. Она означает, что при прорисовке изображения операционная система должна центрировать его по горизонтали, не изменяя размера. Размещение по вертикали в данном случае определяется свойствами контейнера (т.е. типом компоновки). Существуют и другие константы, например `center_vertical`, `left_horizontal` и т.д. Полный список констант можно найти в документации класса `linearLayout.LayoutParams`.
- ✓ **`android:src="@drawable/phone_on"`.** Это свойство является прямым потомком класса `ImageView`. Оно используется для задания файла изображения.

Используемое в данном примере значение `@drawable/phone_on` свойства `src` определено в файле `R.java`. На него можно ссылаться в разметке XML, введя символ `@` перед именем ресурса.

Установка отображаемых ресурсов

Маршрут идентификатора ресурса (например, в атрибуте `src`) записан как `@drawable`, хотя в папке `res` мы записали его во вложенную папку `drawable-mdpi`. Куда подевался суффикс `mdpi`? Не ошиблись ли мы? Нет, не ошиблись. Во время разработки система компоновки Android знает лишь о ресурсах `drawable`. В это время

надстройка ADT ничего не знает об экранах с низкой, средней или высокой разрешающей способностью. И только во время выполнения операционная система Android решает, из какой папки взять ресурс — `ldpi`, `mdpi` или `hdpi`.

Например, если приложение выполняется в устройстве с высокой разрешающей способностью экрана и запрошенный ресурс есть в папке `drawable-hdpi`, операционная система Android извлекает и применяет этот ресурс. Если в этой папке запрошенного ресурса нет, Android извлекает ресурс из папки `drawable-mdpi`, а если и в этой папке его нет, то из папки `drawable-ldpi`, согласно принципу "за неимением гербовой бумаги пишут на простой". Поддержка разных размеров и разрешений экрана — большая и сложная тема, поэтому в данной книге она не рассматривается.

Фрагмент `phone_on` идентифицирует рисунок, который нужно отобразить на экране. Рисунок хранится в файле `phone_on.png`. Согласно принятым в Java соглашениям об именовании расширение `.png` удаляется из идентификатора ресурса в файле `R.java`.

Благодаря надстройке ADT варианты, доступные в данном месте и в данный момент времени, можно увидеть в окне автозавершения кода. В редакторе Eclipse откройте файл `main.xml` и установите курсор ввода в значение атрибута `src` класса `ImageView`. Нажмите комбинацию клавиш <Ctrl+пробел>. Активизируется диалоговое окно автозавершения кода (рис. 4.10). В нем приведены имена ресурсов, которые можно вставить в то место, где находится курсор. Если бы курсор находился на несколько символов правее, в списке отсутствовал бы ресурс `icon`, потому что в подсказке автозавершения учитываются символы слева от курсора. Поэтому при вводе значения можно в любой момент нажать комбинацию клавиш <Ctrl+пробел> и увидеть, что можно вставить в это место.

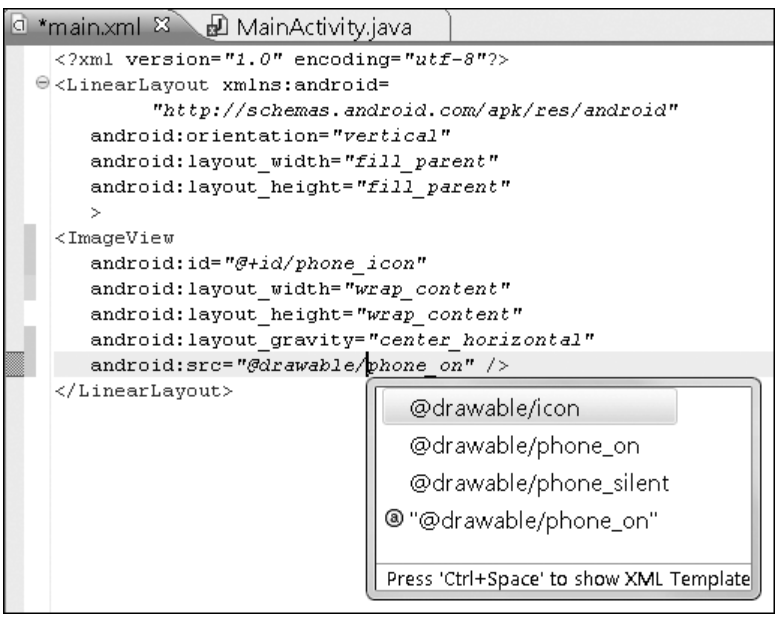


Рис. 4.10. Окно автозавершения кода

Создание значка запуска приложения

Когда приложение установлено, с ним ассоциирован значок запуска, помогающий пользователю найти и запустить приложение. Обычно значок запуска приложения (для краткости часто пишут *значок приложения*) размещается на панели запуска приложений, но он может располагаться и в других местах. В любом случае с его помощью пользователь узнает данное приложение. Когда вы создали приложение Silent Mode Toggle, надстройка ADT автоматически добавила в него стандартный значок Android (рис. 4.11).

По умолчанию стандартный значок добавляется в каждое создаваемое приложение. Но в таком случае какая от него польза? Если все приложения имеют один и тот же значок, станет ли пользователю легче различать их? Совершенно правильно! Пользы от стандартного значка почти никакой. Поэтому вы должны для каждого приложения создать собственный значок, внешний вид которого напоминает о том, что делает данное приложение. Для приложения Silent Toggle Mode я создал значок, изображающий зачеркнутую телефонную трубку (рис. 4.12). Мне кажется, такой значок напоминает об отключенном сигнале. Впрочем, вы можете не согласиться со мной и создать еще более "информативный" значок, как показано в следующем разделе, или выбрать один из многочисленных образцов, загружаемых вместе с Android.

Рис. 4.11. Стандартный значок запуска приложения Android 2.2

Рис. 4.12. Значок приложения Silent Mode Toggle

Создание пользовательского значка приложения

Легко ли создать собственный значок? И да, и нет. С одной стороны, процедура создания значка довольно простая и подробно описана во многих источниках, включая официальное руководство:

```
http://d.android.com/guide/practices/ui_guidelines/icon_design.html
```

Однако, с другой стороны, хороший значок — это произведение искусства. Он должен быть не только эстетически привлекательным при разных разрешениях экрана и в разной обстановке, но и информировать о назначении приложения.

Использование шаблона

В главе 2 вы загрузили пакет инструментов Android SDK, поэтому шаблоны значков и руководство по их созданию есть на вашем жестком диске. Откройте инсталляционную папку Android SDK (см. главу 2) и в ней найдите папку `docs/shareables`. Вы увидите огромное количество архивных файлов `.zip`, содержащих образцы и шаблоны. Откройте шаблон в вашей любимой графической программе и, руководствуясь приведенными в этой же папке информационными материалами, отредактируйте выбранные образцы или используйте готовые значки.

Размеры значков и разрешение экрана

Мобильные устройства оснащены экранами с разными размерами и разрешениями. При создании значка нужно учитывать, как его размер должен зависеть от разрешения экрана. Для каждого разрешения нужно создать отдельный вариант значка определенного размера, чтобы он не выглядел крупнозернистым или вытянутым. Только тогда при любом разрешении экрана значок будет выглядеть эстетически привлекательным.

В табл. 4.4 приведены рекомендуемые размеры значка в пикселях для разных разрешений экрана.

Таблица 4.4. Размеры значков приложений

Разрешение экрана	Размер значка
Низкое, `ldpi`	36×36 px
Среднее, `mdpi`	48×48 px
Высокое, `hdpi`	72×72 px

Добавление значка приложения в проект

Чтобы добавить в проект пользовательский значок запуска приложения, выполните следующие операции.

1. **Переименуйте файл значка. Присвойте ему имя `icon.png`. Это стандартное имя файла значка.**
2. **Перетащите файл значка из проводника Windows в папку `mdpi` на левой панели Eclipse.**

   Программа Eclipse спросит, хотите ли вы переопределить существующий значок `icon.png` (рис. 4.13). Щелкните на кнопке Yes.

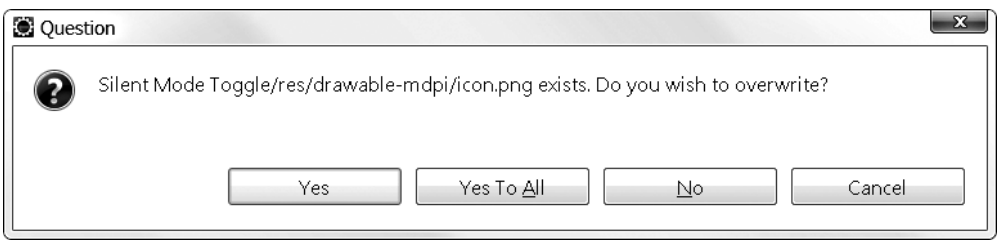


Рис. 4.13. Программа спрашивает, нужно ли переопределить существующий файл значка

Как уже отмечалось, при работе с изображениями рекомендуется создать их разные версии для разных разрешений экрана. Это же справедливо и для значка приложения. Создайте три версии: для высокого, среднего и низкого разрешения экрана, тогда значок будет хорошо выглядеть на любом устройстве. Создание значка рассматривается в официальном сетевом руководстве по Android:

`http://d.android.com/guide/practices/ui_guidelines/icon_design.html`

Вы добавили значок в папку mdpi, но есть еще папки ldpi и hdpi. Поэтому вы должны создать еще две версии значка (т.е. еще два файла .png) для высокого и низкого разрешений экрана. Создав эти два файла, перетащите их в папки hdpi и ldpi соответственно, как описано выше для папки mdpi.

Если не скопировать значки с высоким и низким разрешением в соответствующие папки, то при высоком или низком разрешении экрана для приложения будет отображаться стандартный значок Android, показанный на рис. 4.11, а при среднем разрешении — созданный вами значок.

На первый взгляд, это противоречит правилам автоматического выбора изображений операционной системой. Согласно им, если изображения с данным разрешением нет, операционная система применяет изображение с другим, ближайшим разрешением. Но на самом деле противоречия нет. В папках hdpi и ldpi всегда есть стандартные значки приложения icon.png, поэтому операционная система применяет их, не замечая, что они не похожи на значок в папке mdpi.

Добавление кнопки

Кнопка, размещенная на экране, является виджетом. *Виджет* — это интерактивный элемент интерфейса, служащий для взаимодействия пользователя с мобильным устройством. Все виджеты являются представлениями, поскольку наследуют класс View. От других представлений виджеты отличаются свойством интерактивности, т.е. тем, что пользователь не только видит их, но и выполняет с их помощью определенные действия. Операционная система Android оснащена большим количеством виджетов разных типов, включая кнопки, флажки, переключатели, раскрывающиеся списки, текстовые поля и т.д., благодаря которым можно быстро создать графический интерфейс пользователя. Существуют и более сложные виджеты, чем перечисленные выше, например поле выбора даты, часы, элемент управления зумированием и др.

Виджеты предоставляют события пользовательского интерфейса, информирующие приложение о том, что пользователь выполнил некоторую операцию с данным виджетом, например щелкнул на кнопке.

В приложение Silent Toggle Mode нужно добавить кнопку, чтобы пользователь мог переключить режим звонка в мобильном телефоне. Чтобы добавить кнопку на экран, введите в файле main.xml следующий код после кода элемента ImageView.

```
<Button
   android:layout_height="wrap_content"
   android:text="Переключить режим звонка"
   android:layout_gravity="center_horizontal"
   android:layout_width="wrap_content"
   android:id="@+id/toggleButton" >
</Button>
```

Вы добавили в интерфейс кнопку с идентификатором ресурса toggleButton, с помощью которого будете ссылаться на кнопку в коде Java в следующей главе.

Атрибутам высоты и ширины кнопки присвоено одинаковое значение wrap_content, которое извещает операционную систему Android о том, что виджет должен занимать на экране столько пространства, сколько ему необходимо. В данном случае ширина кнопки определяется длиной фразы Переключить режим звонка, а высота кнопки — высотой шрифта. С помощью свойства layout_gravity виджет центрируется по горизонтали. Текст, отображаемый на кнопке, задается в атрибуте text.

Сейчас файл `main.xml` должен содержать приведенный ниже код. Глядя на него, хорошо видно, что на экране отображаются два элемента интерфейса: `ImageView` и `Button`.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
        "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <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:layout_height="wrap_content"
                android:text="Переключить режим звонка"
                android:layout_gravity="center_horizontal"
                android:layout_width="wrap_content"
                android:id="@+id/toggleButton" >
    </Button>
</LinearLayout>
```

Приложение в режиме конструктора

Теперь посмотрим, как созданный нами графический интерфейс приложения выглядит в окне конструктора. Откройте на правой панели Eclipse файл `main.xml` и в нижней части экрана щелкните на вкладке Graphical Layout (Графическая компоновка). На правой панели Eclipse будет открыто окно конструктора (рис. 4.14).

Визуальная среда разработки, предоставляемая надстройкой ADT

Окно конструктора создается надстройкой ADT, добавленной в программу Eclipse. Режим конструктора может иметь много разных конфигураций. По умолчанию используется конфигурация ADP1 (Android Development Phone 1 — разработка телефона Android, версия 1). Это был первый шаблон мобильного приложения, представленный компанией Google. При желании вы и сейчас можете купить устройство ADP1, чтобы протестировать на нем приложение на основе данного шаблона. Щелкнув на кнопке Create (Создать), можно выбрать другой шаблон. Например, в шаблоне Querty на экране отображается клавиатура. Над окном конструктора находится ряд раскрывающихся списков (см. рис 4.14), с помощью которых можно выбрать тему, ориентацию, разрешение и другие параметры шаблона. Например, для шаблона ADP1 раскрывающийся список доступных ориентаций предоставляет три значения.

- ✓ **Landscape, closed** (Альбомная, закрытая). Для отображения интерфейса используется альбомная ориентация. Клавиатура скрыта.
- ✓ **Potrait** (Портретная). Используется портретная ориентация.
- ✓ **Landscape, open** (Альбомная, открытая). Альбомная ориентация. На экране отображается клавиатура.

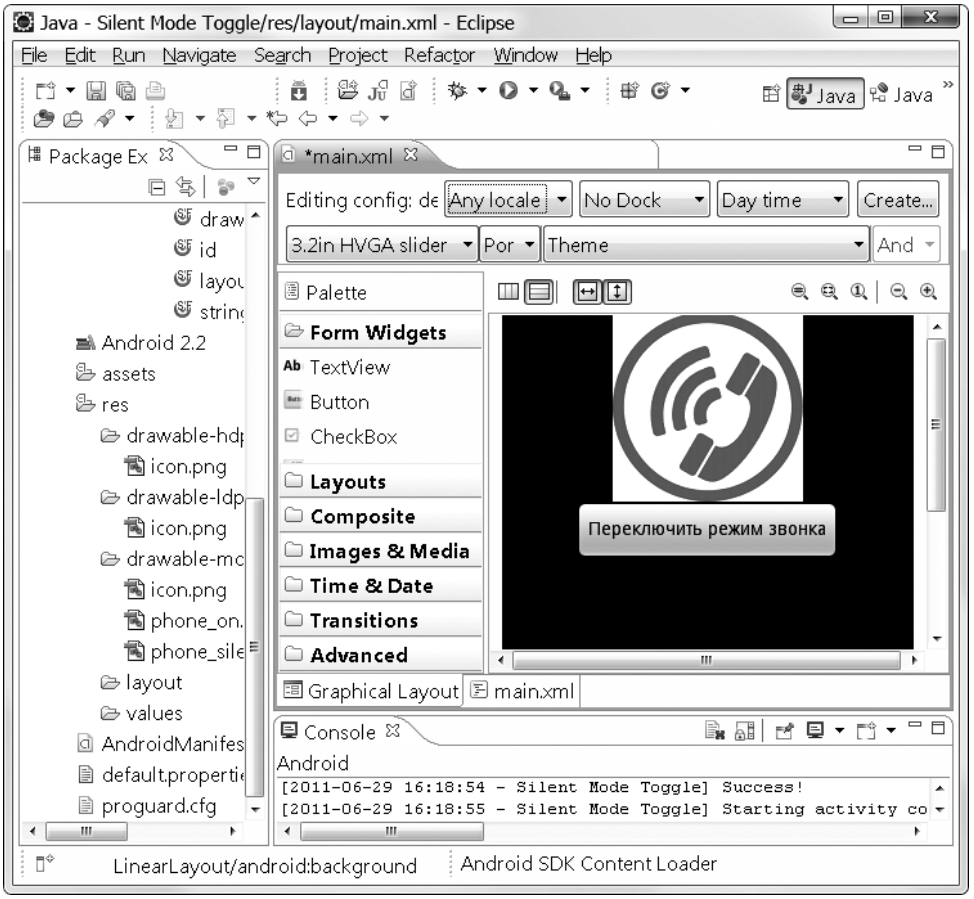


Рис. 4.14. Приложение Silent Mode Toggle в окне конструктора

Изменение цвета фона

Цвет фона у нас черный, а изображение белое. Как-то это некрасиво. Давайте сделаем цвет фона белым, чтобы изображение не выглядело как грубый квадрат, а плавно вписывалось в окружающую обстановку. Чтобы изменить цвет фона, выполните следующие операции.

1. **В нижней части окна щелкните на вкладке `main.xml`, чтобы переключить окно из режима конструктора в режим кодирования.**
2. **Добавьте в контейнер `LinearLayout` атрибут `background` следующим образом.**

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

3. **Определение контейнера теперь должно выглядеть, как показано ниже.**

   ```
   <LinearLayout xmlns:android=
           "http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="fill_parent"
   ```

```
android:layout_height="fill_parent"
android:background="#ffffff"
>
```

4. **Сохраните файл `main.xml`.**
5. **Щелкните на вкладке Graphical Layout, чтобы отобразить компоновку в режиме конструктора.**

   Теперь правая панель Eclipse должна выглядеть, как показано на рис. 4.15.

   Атрибуту фона присвоено значение `#ffffff`, что означает непрозрачный белый цвет. Можно задать любой цвет. Например, значение `#ff0000` определяет красный фон. С помощью ресурсов (т.е. папки `res`) можно установить не одноцветный фон, а фоновое изображение. Однако почти всегда лучше одноцветный фон, потому что фоновое изображение редко удается сделать красивым. С одной стороны, его портят размещенные на нем виджеты, а с другой — оно само затрудняет просмотр виджетов, потому что делает фон пестрым.

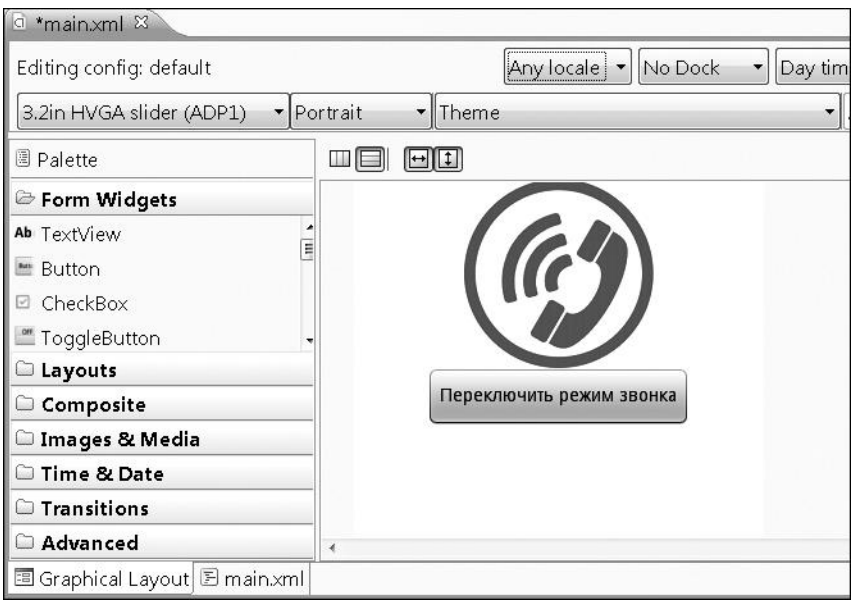


Рис. 4.15. На белом фоне рисунок и кнопка выглядят лучше, чем на черном

Итак, мы создали графический интерфейс приложения Silent Mode Toggle. Но это всего лишь картинка, а не работающее приложение. В следующей главе мы “вдохнем в него жизнь”, т.е. запрограммируем операции, которые оно должно выполнять.

Глава 5

Кодирование приложения

В этой главе...

- Что такое деятельность
- Создание деятельности
- Работа с базовыми классами Android
- Установка приложения
- Переустановка приложения
- Отладка
- Выход за границы приложения

Вам, конечно, не терпится побыстрее приступить к созданию кода приложения! Если бы я был на вашем месте, то испытывал бы такие же чувства. В этой главе мы займемся кодированием, однако немного потерпите: прежде чем с головой погрузиться в пучину байтов и битов, нужно кое-что узнать о деятельностях.

Что такое деятельность

В операционной системе Android *деятельность* (activity) — это программная реализация единой, конкретной операции, которую может выполнить пользователь. Например, деятельность может представлять на экране список элементов меню, доступных для пользователя, или отображать фотографии с заголовками. Приложение Android может состоять из одной деятельности, но большинство приложений состоит из нескольких. Деятельности могут взаимодействовать друг с другом, благодаря чему они выглядят как одно приложение, однако фактически независимы друг от друга. Деятельность — фундаментальный элемент инфраструктуры Android и жизненного цикла приложения, поэтому способ запуска деятельностей и принципы их взаимодействия — важная часть модели приложений. С программной точки зрения каждая деятельность — это реализация базового класса `Activity`.

Почти все деятельности взаимодействуют с пользователем, поэтому класс `Activity` создает окно, в котором можно разместить пользовательский интерфейс. Чаще всего деятельности представлены в полноэкранном режиме, однако в случае необходимости можно разместить деятельность в плавающем окне или внедрить в другую деятельность (т.е. создать группу деятельностей).

Методы, стеки и состояния

Почти все деятельности реализуют два следующих метода.

- ✓ **`onCreate()`.** В этом методе вы инициализируете деятельность и, что еще важнее, задаете компоновку, применяемую деятельностью при размещении ресурсов.
- ✓ **`onPause()`.** В этом методе вы кодируете операции, которые должны быть выполнены, когда пользователь прекращает работу с деятельностью. Любые изменения, выполненные пользователем, должны быть зафиксированы (т.е. сохранены или обработаны каким-либо иным способом) именно в этом методе.

Операционная система манипулирует деятельностями как элементами *стека деятельностей*. Когда создается новая деятельность, она размещается на вершине стека и становится текущей (т.е. выполняющейся). Предыдущая текущая деятельность при этом размещается в стеке на ступеньку ниже. После этого она никогда не станет текущей, пока активна новая текущая деятельность.

Чрезвычайно важно понимать, как деятельности работают "за кулисами". Понимание этого не только является фундаментом концепции операционной системы Android, но и жизненно важно для отладки приложений, когда во время выполнения происходят странные, непредсказуемые события.

Деятельность имеет четыре состояния, перечисленные в табл. 5.1.

Таблица 5.1. Четыре основных состояния деятельности

Состояние	Описание
Активная (другие названия — текущая или выполняющаяся)	На экране деятельность находится на переднем плане, а в стеке деятельностей — на его вершине
Пауза	Деятельность потеряла фокус, но все еще видима (возможно, фокус принадлежит деятельности, занимающей не весь экран, или прозрачной деятельности). В режиме паузы деятельность остается "живой", т.е. она сохраняет свои данные и остается подключенной к менеджеру окно, который управляет окнами операционной системы Android. Однако учитывайте, что при нехватке памяти (а это легко может произойти в портативном устройстве) операционная система Android может уничтожить деятельность, находящуюся в режиме паузы
Остановлена	Когда деятельность полностью закрывается другими окнами, Android останавливает ее. В остановленной деятельности сохраняются все данные, она лишь не видна пользователю. При нехватке памяти вероятность уничтожения выше, чем в режиме паузы
Создается или возобновляется	Операционная система переключила в режим паузы или остановила деятельность. Операционная система либо отбирает у нее ресурсы памяти, либо уничтожает ее процесс. Когда такая деятельность видна пользователю, она может быть возобновлена путем повторного запуска и восстановления прежнего состояния

Жизненный цикл деятельности

Лучше один раз увидеть, чем сто раз услышать, поэтому одна диаграмма переключения состояний деятельности (рис. 5.1) объяснит ее жизненный цикл лучше, чем десяток страниц текстового описания.

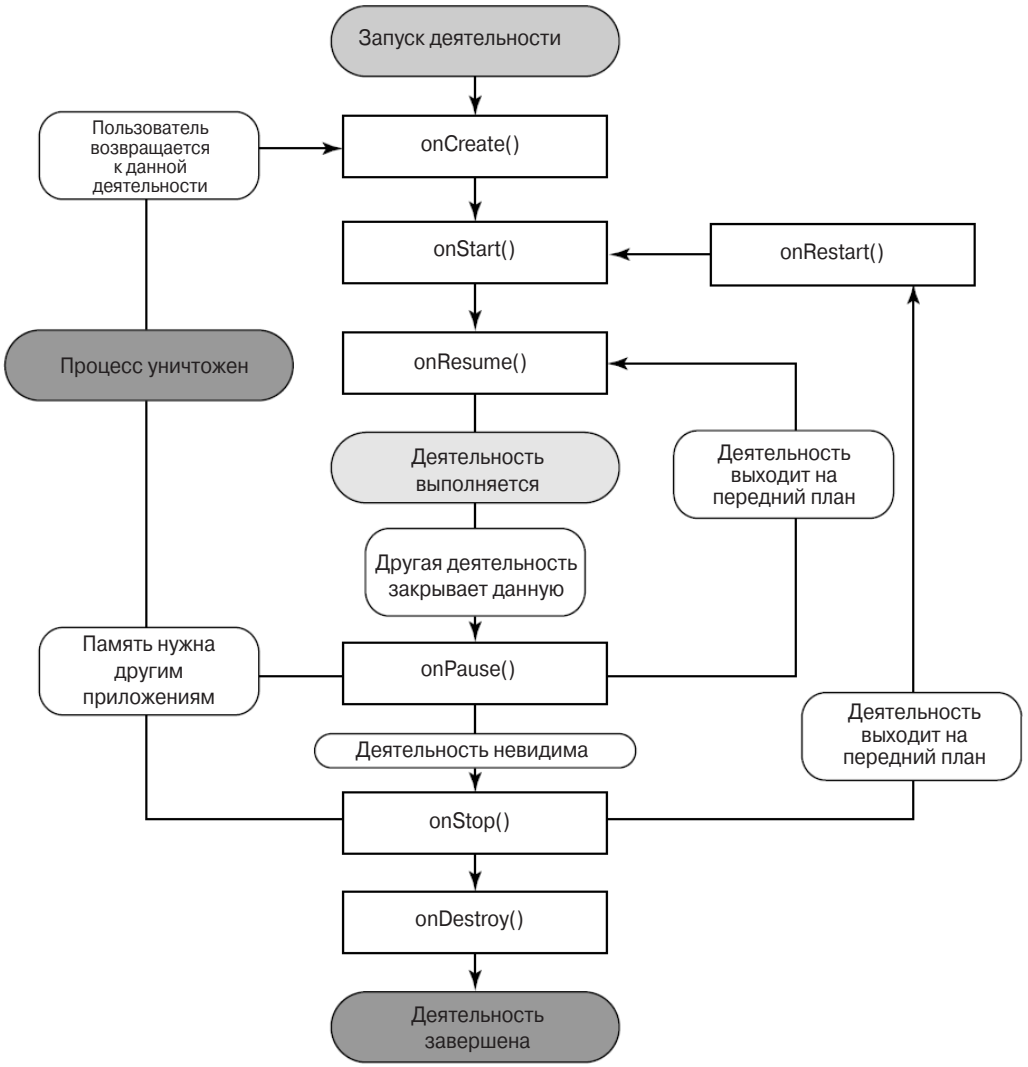


Рис. 5.1. Жизненный цикл деятельности

Прямоугольники обозначают кодируемые вами методы обратного вызова, которые реагируют на события данной деятельности. Затененные овалы — это главные состояния, в которых может пребывать деятельность.

Основные петли жизненного цикла

Наиболее важны три следующие петли.

- ✓ **Весь жизненный цикл** находится между первым вызовом метода `onCreate()` и последним вызовом `onDestroy()`. В методе `onCreate()` деятельность настраивает все свои глобальные параметры, а в методе `onDestroy()` — освобождает все оставшиеся ресурсы. Например, если вы создаете поток для загрузки файла из Интернета в фоновом режиме, то в методе `onCreate()` поток инициализируется, а в методе `onDestroy()` — завершается.
- ✓ **Видимое состояние** имеет место между методами `onStart()` и `onStop()`. На протяжении времени между вызовами этих методов пользователь видит деятельность на экране, хотя она не обязательно находится на переднем плане и доступна для взаимодействия пользователя с ней (например, когда пользователь взаимодействует с другим диалоговым окном). Между этими двумя методами можно поддерживать ресурсы, необходимые для отображения и выполнения деятельности. Например, можно создать обработчик события для отслеживания состояния мобильного телефона. Когда состояние телефона изменяется, обработчик может известить деятельность об этом и обеспечить этим правильную реакцию на изменение состояния. Обработчик настраивается в методе `onStart()`, а в методе `onStop()` уничтожаются ресурсы, используемые деятельностью (чтобы освободить память). Методы `onStart()` и `onStop()` могут вызываться многократно, когда деятельность становится видимой или скрывается от пользователя.
- ✓ **Деятельность находится на переднем плане.** Это состояние начинается в момент вызова метода `onResume()` и заканчивается методом `onPause()`. На протяжении времени между вызовами этих методов деятельность прорисовывается поверх всех других деятельностей и взаимодействует с пользователем. Обычно деятельность проходит этап между вызовами методов `onResume()` и `onPause()` многократно, например, когда мобильное устройство переводится в спящий режим или новая деятельность обрабатывает некоторое событие. Следовательно, коды этих методов должны быть простыми и выполняться быстро.

Методы деятельности

Весь жизненный цикл деятельности заключен в перечисленных ниже методах. Все эти методы можно (и нужно) переопределять, размещая в них пользовательский код. Все деятельности реализуют метод `onCreate()` для инициализации процесса и могут реализовать метод `onPause()` для очистки системы. При реализации перечисленных ниже методов нужно всегда вызывать конструктор базового класса.

```
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();
}
```

Продвижение деятельности по жизненному циклу

В общем случае деятельность продвигается по своему жизненному циклу следующим образом.

- ✓ **`onCreate()`**. Этот метод вызывается при первом создании деятельности. В нем программист обычно инициализирует большинство переменных деятельности на уровне класса. После `onCreate()` всегда вызывается `onStart()`. Метод `onCreate()` недоступен для уничтожения. Следующий — `onStart()`.
- ✓ **`onRestart()`**. Вызывается после остановки деятельности и перед ее повторным запуском. После `onRestart()` всегда вызывается `onStart()`. Недоступен для уничтожения. Следующий — `onStart()`.
- ✓ **`onStart()`**. Вызывается, когда деятельность становится видимой для пользователя. После него вызывается `onResume()`, если деятельность выводится на передний план, или `onStop()`, если деятельность скрывается от пользователя. Недоступен для уничтожения. Следующий — `onResume()` или `onStop()`.
- ✓ **`onResume()`**. Вызывается, когда деятельность становится доступной для взаимодействия с пользователем. В этот момент деятельность помещается на вершину стека деятельностей. Недоступен для уничтожения. Следующий — `onPause()`.
- ✓ **`onPause()`**. Вызывается, когда система собирается возобновить предыдущую деятельность или когда пользователь переходит к другой части системы, например нажав кнопку перехода к главному экрану. Данный метод обычно используется для сохранения данных, которые должны быть постоянными. Если после этого деятельность вновь выводится на передний план, следующим вызывается метод `onResume()`, а если деятельность должна стать невидимой — то метод `onStop()`. Доступен для уничтожения. Следующий — `onResume()` или `onStop()`.
- ✓ **`onStop()`**. Вызывается, когда деятельность перестает быть видимой для пользователя вследствие того, что ее закрыла другая, только что возобновленная деятельность. Это может произойти, когда запускается другая или возобновляется предыдущая деятельность. Новая деятельность помещается на вершину стека деятельностей. Если данная деятельность становится доступной для взаимодействия с пользователем, следующим вызывается метод `onRestart()`, а если деятельность завершается, то вызывается метод `onDestroy()`. Доступен для уничтожения. Следующий — `onRestart()` или `onDestroy()`.

- ✓ **`onDestroy()`**. Вызывается в самом конце жизненного цикла, перед уничтожением деятельности. Вызов может быть порожден одним из двух событий: либо в коде был вызван метод `finish()`, либо система временно уничтожила деятельность для получения нужного ей объема оперативной памяти. Выяснить, какая из этих двух причин имела место, можно с помощью метода `isFinished()`. Этот метод часто используется в методе `onPause()` для выяснения, что происходит с деятельностью: она находится в режиме паузы или уничтожена. Доступен для уничтожения. Следующего нет.

В конце описания каждого метода приведена информация, доступен ли он для уничтожения и какой метод запускается после данного. Что такое "следующий", видимо, понятно без комментариев, а вот доступность для уничтожения может показаться загадочным свойством. Но ничего загадочного в нем нет. Дело в том, что метод, отмеченный как "доступный для уничтожения", может быть прерван и уничтожен операционной системой Android в любой момент без предупреждения (например, когда зачем-то понадобилась память, занимаемая деятельностью). Поэтому для окончательной очистки приложения и сохранения постоянных данных (например, тех, что были введены пользователем) следует использовать метод `onPause()`.

Обнаружение изменения конфигурации

Еще одно замечание о жизненном цикле деятельности, и мы, наконец, сможем приступить к кодированию. При разработке приложения для мобильного устройства нужно учитывать, что его конфигурация может измениться в любой момент. Например, может измениться ориентация экрана, язык ввода, указательное устройство и пр. Изменение конфигурации влечет уничтожение деятельности путем ее прохождения по стандартному маршруту: `onPause()` ⇨ `onStop()` ⇨ `onDestroy()`. После вызова метода `onDestroy()` операционная система создает новый экземпляр деятельности. Это необходимо потому, что ресурсы, файлы компоновки, активы и другие компоненты деятельности могут измениться. Например, когда пользователь поворачивает устройство и портретный режим отображения сменяется альбомным, внешний вид пользовательского интерфейса радикально изменяется.

Жизненный цикл деятельности — большая и сложная тема. Я ознакомил вас лишь с базовыми понятиями, необходимыми для понимания примеров данной книги. Когда вы прочитаете книгу до конца, рекомендую более подробно ознакомиться с этапами жизненного цикла деятельности с помощью ресурсов Интернета.

Создание деятельности

Фактически вы уже создали первую в своей жизни деятельность, читая главу 3 и работая с приложением Silent Mode Toggle. Эта деятельность называлась `MainActivity` и находилась в файле `MainActivtity.java`. Откройте этот файл в Eclipse.

Начнем с метода `onCreate`

Как указано выше, входной точкой приложения служит метод `onCreate()`. В коде файла `MainActivity.java` уже есть реализация этого метода. С этого момента вы начнете писать код. Сейчас ваш код должен выглядеть так.

```
public class MainActivity extends Activity {
   /** Вызывается при создании деятельности */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }
}
```

Код инициализации вы напишете непосредственно под вызовом метода `setContentView()`.

Обратите внимание на следующую строку:

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

Без нее приложение не будет работать. Префикс `super` задает вызов метода `onCreate()`, определенного в базовом классе и настраивающего параметры класса `MainActivity`. Если не включить эту строку в код, операционная система сгенерирует ошибку времени выполнения. Поэтому никогда не забывайте включить в свой метод `onCreate()` вызов метода `onCreate()` базового класса.

Объект `Bundle`

В приведенном выше коде обратите внимание на такой аргумент:

```
Bundle savedInstanceState
```

Объект `Bundle` (пучок, связка) содержит значения, связывающие строковые ключи с хранимыми типами деятельности. Они предоставляют способ передачи информации между разными компоновками экрана и деятельностями. Можно разместить нужные типы в объекте `Bundle`, а затем извлечь их из этого объекта в целевой деятельности. Более подробно данная тема рассматривается в части III, в которой вы создадите приложение, напоминающее о том, что нужно сделать.

Отображение пользовательского интерфейса

По умолчанию в деятельности не заложено никакой информации о том, как выглядит ее интерфейс. Это может быть простая форма, позволяющая пользователю вводить и сохранять информацию, или компьютерная игра, отображающая движущиеся объекты. Как разработчик, вы должны сообщить деятельности, какую компоновку она должна загрузить и применить.

Чтобы операционная система отобразила пользовательский интерфейс на экране, нужно установить представление содержимого деятельности. Это делается с помощью следующей инструкции:

```
setContentView(R.laout.main)
```

Аргумент `R.layout.main` ссылается на файл `main.xml`, расположенный в папке `res/layouts` и определяющий компоновку пользовательского интерфейса.

Обработка действий пользователя

Приложение Silent Mode Toggle не обладает богатым набором средств взаимодействия с пользователем. Фактически оно имеет лишь одно такое средство — кнопку. Нажимая эту кнопку на экране, пользователь переключает режим звонка между громким и бесшумным.

Чтобы приложение отреагировало на прикосновение пользователя к кнопке, нужно зарегистрировать *приемник события*. Событием в данном случае является нажатие кнопки пользователем. Операционная система Android поддерживает около десятка стандартных типов событий. Два наиболее популярных — событие прикосновения к элементу интерфейса на экране и событие клавиатуры.

События клавиатуры

Операционная система генерирует событие клавиатуры при нажатии любой клавиши. С помощью событий клавиатуры можно запрограммировать для приложения *горячие клавиши*. Например, при нажатии клавиш <Alt+E> приложение обычно переключается в режим редактирования. Но чтобы приложение отреагировало на нажатие этих клавиш, в нем должен быть зарегистрирован приемник данного события. В примерах данной книги события клавиатуры не используются, но в будущем вы наверняка будете разрабатывать приложения, в которых без них не обойтись. Поэтому вам полезно будет знать, что для перехвата события клавиатуры нужно переопределить метод `onKeyDown()`, как показано ниже.

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
   // Здесь введите код, выполняемый при нажатии клавиши
   return super.onKeyDown(keyCode, event);
}
```

События прикосновения

Операционная система генерирует событие прикосновения, когда пользователь прикасается к виджету на экране. Каждое прикосновение распознается как щелчок, поэтому термины *прикосновение* и *щелчок* — синонимы. Ниже приведен неполный список виджетов, реагирующих на прикосновения:

- ✓ `Button`;
- ✓ `ImageButton`;
- ✓ `EditText`;
- ✓ `Spinner`;
- ✓ `ListItemRow`;
- ✓ `MenuItem`.

Все представления могут реагировать на прикосновение, но у некоторых виджетов свойство `Clickable` (Чувствительный к щелчку) по умолчанию равно `false`. Можете переопределить его в файле компоновки (изменив значение атрибута `clickable`) или в коде приложения (вызвав метод `setClickable()`). Тогда виджет будет реагировать на прикосновения.

Создание обработчика события

Чтобы приложение Silent Mode Toggle реагировало на прикосновение пользователя, нужно запрограммировать в нем обработку события щелчка на кнопке.

Код обработчика

В окне редактора введите код, приведенный в листинге 5.1. Данный код иллюстрирует реализацию обработчика щелчка на кнопке `toggleButton`. Можете либо ввести только код для кнопки, либо переопределить весь метод `onCreate()`.

Листинг 5.1. Исходный код класса деятельности с приемником щелчка, установленным по умолчанию

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

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

В этом коде используется метод `findViewById()`, доступный для всех деятельностей Android. Данный метод позволяет найти в компоновке деятельности любое представление и сделать с ним что-нибудь. Метод всегда возвращает класс `View` (Представление), который нужно привести к соответствующему типу перед его использованием. В приведенной ниже строке возвращенный объект `View` приводится к классу `Button` (Кнопка), производному от класса `View`.

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

Сразу после этого можно начать установку обработчика события.

Код обработки события встроен после извлечения объекта `Button` из компоновки. Установка обработчика выполняется путем создания приемника `View.OnClickListener`, который содержит метод `onClick()`. Обработчиком события служит метод `onClick()`, вызываемый операционной системой при нажатии кнопки пользователем. В тело метода `onClick()` нужно добавить код, переключающий режим звонка.

Если тип представления в файле компоновки отличается от типа, к которому вы пытаетесь привести представление (например, в файле компоновки применяется тип `ImageView`, а вы пытаетесь привести представление к типу `ImageButton`), будет сгенерирована ошибка времени выполнения. Поэтому убедитесь в том, что представление приводится к правильному типу.

При вводе кода метода `onCreate()` в окне редактора в первый раз под спецификатором `Button` может появиться красная волнистая линия и окно, сообщающее об ошибке (рис. 5.2). Программа Eclipse говорит вам, что она не знает, что такое `Button`. Окно сообщения появляется либо в момент ввода, либо при наведении указателя на `Button`. Но даже если оно не появилось, красной волнистой линии вам должно быть достаточно, чтобы понять, что в программе есть ошибка.

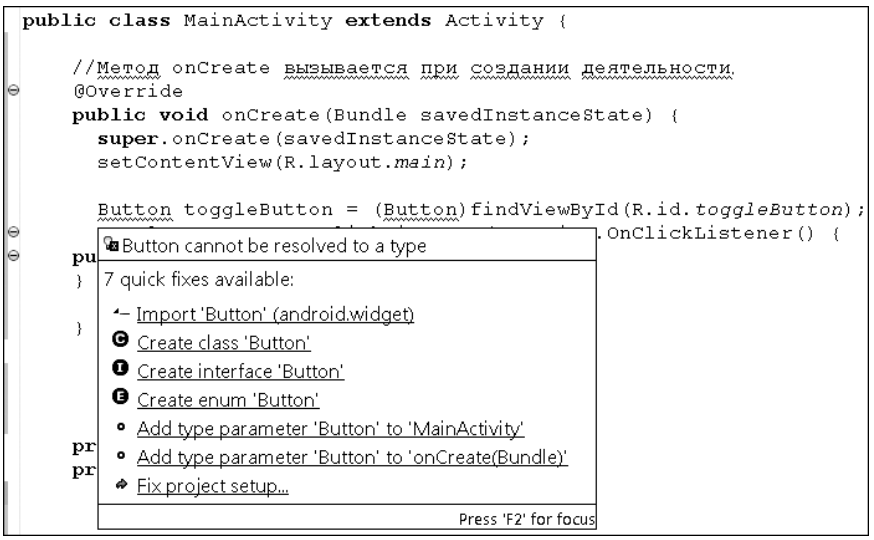


Рис. 5.2. Программа Eclipse сообщает о том, что не может найти класс `Button`

Программа не понимает, что вы хотите сделать, и предлагает ряд способов исправления ошибки. Правильным оказался только первый способ — добавление оператора импорта в верхней части файла.

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

Данная инструкция информирует Eclipse о том, что `Button` — это класс, определенный в пакете `android.widget`. Таким же образом должны быть импортированы библиотеки всех элементов интерфейса, используемых в приложении.

Разрабатывая более сложные приложения и включая в них многие виджеты, вы обнаружите, что список инструкций `import` становится чересчур длинным. Чтобы сделать его короче, нужно учитывать, что не обязательно писать отдельную инструкцию для каждого виджета. Существует простой способ импорта всего содержимого пакета. Для этого достаточно поставить звездочку после имени пакета:

```
import android.widget.*
```

Эта инструкция приказывает Eclipse включить в деятельность все виджеты пакета `android.widget`.

Перенос кода в метод

По мере усложнения приложения код становится все более громоздким и его все тяжелее читать. Чтобы упростить задачу, рекомендуется извлечь код кнопки в отдельный метод, вызываемый в теле обработчика `onCreate()`. Для этого создайте закрытый метод `setButtonClickListener()`, содержащий код кнопки. Вызов этого метода поместите в метод `onCreate()`. Новый код показан в листинге 5.2.

Листинг 5.2. Код установки приемника события извлечен в отдельный метод

```
public class MainActivity extends Activity {
   /** Этот метод вызывается при создании деятельности */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      setButtonClickListener();                              16
   }

   private void setButtonClickListener() {                   19
      Button toggleButton =
               (Button)findViewById(R.id.toggleButton);
      toggleButton.setOnClickListener(
               new View.OnClickListener() {
         public void onClick(View v) {
            // Код обработки события щелчка
         }
      });
   }
}
```

- ✓ **16.** В этой строке расположен вызов метода, содержащего код установки приемника события.
- ✓ **19.** В этой строке находится заголовок метода, содержащего код установки приемника.

Теперь можно запрограммировать реакцию на событие нажатия кнопки, добавив нужный код в тело метода `onClick()`, который называется *обработчиком события*.

Работа с базовыми классами Android

Наконец-то мы переходим к самому интересному — базовым классам инфраструктуры Android! Деятельности, представления и виджеты — жизненно важная часть Android, но это всего лишь рабочие инструменты, доступные (в той или иной форме) в любой современной операционной системе. Однако Android — это операционная система, предназначенная для мобильных устройств, что существенно влияет на специфику ее инфраструктуры.

В этом разделе мы запрограммируем проверку состояния звонка, чтобы выяснить, в каком режиме он пребывает — громком или бесшумном. Это позволит нам переключать режим звонка путем нажатия кнопки.

Программное управление звонком

Получить доступ к звонку мобильного телефона можно с помощью базового класса `AudioManager` операционной системы, который отвечает за управление состояниями звонка. В рассматриваемом примере деятельности класс `AudioManager` часто используется, поэтому лучше инициализировать его с самого начала — в методе `onCreate()`. Не забывайте, что все параметры приложения лучше всего инициализировать в методе `onCreate()`.

Сначала создайте закрытую переменную уровня класса `AudioManager` с именем `mAudioManager`. Введите ее в верхней части файла класса, непосредственно после строки объявления класса (листинг 5.3).

Листинг 5.3. Добавление переменной `mAudioManager`

```
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.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) {
         // Код обработки прикосновения к кнопке
         }
      });
   }
}
```

Ниже приведено краткое описание отмеченных строк.

- ✓ **4.** Оператор `import` подключает нужный пакет. Поэтому в приложении всегда можно использовать класс `AudioManager`.
- ✓ **11.** Объявление закрытой переменной уровня класса `mAudioManager`. Эта переменная видна во всем классе, поэтому ее можно использовать в других частях деятельности.
- ✓ **20.** Инициализация переменной `mAudioManager` путем обращения к системной службе посредством вызова метода `getSystemService()`, который определен в базовом классе `Activity`.

Наследуя класс `Activity`, вы получаете все средства деятельностей. В данном примере вы получаете право вызывать метод `getSystemService()`, который возвращает базовый класс `Object`. Следовательно, необходимо привести `Object` к типу запрашиваемой службы.

Метод `getSystemService()` возвращает все доступные системные службы, которые могут понадобиться при работе с деятельностью. Описание всех служб можно найти в описании класса `Context`, приведенном в документации Java по адресу `http://d.android.com/reference/android/content/Context.html`. Ниже перечислены наиболее популярные системные службы:

- ✓ аудиослужба;
- ✓ служба глобального позиционирования;
- ✓ служба оповещения.

Переключение режима звонка с помощью объекта `AudioManager`

Теперь у нас есть экземпляр `AudioManager` на уровне класса, поэтому мы можем проверять состояние звонка и переключать его режим. Эти операции выполняет код, приведенный в листинге 5.4. Все новые инструкции отмечены полужирным шрифтом.

Листинг 5.4. Добавление средств переключения режима звонка

```
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 {
   private AudioManager mAudioManager;
   private boolean mPhoneIsSilent;                          14

   @Override
   public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.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
       // Переключение в режим громкого звонка
       mAudioManager.setRingerMode(
                        AudioManager.RINGER_MODE_NORMAL);
       mPhoneIsSilent = false;
    } else {
       // Переключение в бесшумный режим
       mAudioManager
          .setRingerMode(AudioManager.RINGER_MODE_SILENT);
          mPhoneIsSilent = true;
    }
    // Переключение пользовательского интерфейса
    toggleUi();                                                44
    }
   });
}

/**
* Проверка состояния телефона
*/
    private void checkIfPhoneIsSilent() {                      53
       int ringerMode = mAudioManager.getRingerMode();
       if (ringerMode ==
                       AudioManager.RINGER_MODE_SILENT) {
          mPhoneIsSilent = true;
       } else {
          mPhoneIsSilent = false;
       }
    }

       /**
       * Переключение рисунка
       */
       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();
}
```

В приложение добавлено много новых инструкций. Ниже приведено краткое описание каждого нового фрагмента кода.

- ✓ **14.** Объявление булевой переменной `mPhoneIsSilent` уровня класса, которая отслеживает текущее состояние звонка.
- ✓ **23.** Вызов метода `checkIfPhoneIsSilent()` для инициализации переменной `mPhoneIsSilent`. По умолчанию эта переменная равна `false`, что может быть неправильно, если включен бесшумный режим звонка. Следовательно, чтобы знать, что произойдет при переключении режима звонка (он станет громким или бесшумным), нужно правильно инициализировать переменную `mPhoneIsSilent`.
- ✓ **25.** Код обработки щелчка на кнопке перемещен в нижнюю часть метода `onCreate()`, потому что он зависит от инициализации переменной `mPhoneIsSilent`. При неправильной инициализации или вообще без нее, скорее всего, ничего плохого не произойдет, потому что на первой же итерации переменная `mPhoneIsSilent` примет правильное значение. Однако лучше, чтобы код был правильно организован, тогда с ним будет легче работать.
- ✓ **32.** Код между строками 32 и 44 обрабатывает событие прикосновения пользователя к кнопке. С помощью переменной `mPhoneIsSilent` уровня класса этот код проверяет, включен ли громкий звонок в данный момент. Если он не включен, код передает управление первому блоку `if`, который изменяет режим звонка на `RINGER_MODE_NORMAL`, что в свою очередь приводит к включению звонка. При этом переменная `mPhoneIsSilent` получает значение `false` и сохраняет его до следующей активизации данного кода. Если звонок громкий, управление передается блоку `else`. Код блока `else` переключает режим звонка с текущего состояния в состояние `RINGER_MODE_SILENT`, что приводит к отключению звонка. Кроме того, блок `else` присваивает переменной `mPhoneIsSilent` значение `true`, которое сохраняется до следующего прикосновения пользователя к кнопке.

- ✓ **44.** Метод `toggleUi()` изменяет пользовательский интерфейс, чтобы предоставить пользователю визуальную информацию о текущем режиме звонка. При каждом изменении режима звонка нужно вызвать метод `toggleUi()`.
- ✓ **53.** Метод `checkIfPhoneIsSilent()` извлекает из устройства текущий режим звонка, который используется для инициализации переменной `mPhoneIsSilent` на уровне класса в методе `onCreate()`. Без проверки текущего режима приложение не смогло бы узнать, в каком состоянии находится звонок в объекте `AudioManager`. Если телефон находится в бесшумном режиме, переменной `mPhoneIsSilent` присваивается значение `true`, в противном случае — значение `false`.
- ✓ **66.** Метод `toggleUi()` изменяет объект `ImageView`, созданный в предыдущей главе, в зависимости от текущего режима звонка. Если звонок громкий, на экране отображается изображение, показанное на рис. 4.4, а если звонок находится в бесшумном режиме — то изображение на рис. 4.5. В главе 4 оба эти изображения были помещены в папку ресурсов. Экземпляр `ImageView` ищется в компоновке. После выяснения текущего режима представление обновляется путем извлечения нужного изображения из `getResources().getDrawable()` и вызова метода `setImageDrawable()` через объект `ImageView`. Этот метод обновляет изображение, выведенное на экране в элементе `ImageView`.
- ✓ **84.** Выше я указывал на важность понимания жизненного цикла деятельности. В данном примере вы имеете возможность увидеть его на практике. Метод `onResume()` переопределен таким образом, чтобы он мог правильно идентифицировать свое текущее состояние. Не слишком ли много состояний для двух режимов? Не достаточно ли того, что состояние телефонного звонка отслеживается переменной `mPhoneIsSilent`? Но она отслеживает состояние на уровне класса. Пользователь тоже должен знать, в каком состоянии в данный момент находится телефон. Поэтому метод `onResume()` вызывает метод `toggleUi()`, который переключает пользовательский интерфейс. Метод `onResume()` вызывается после `onCreate()`, поэтому в методе `toggleUi()` можно полагаться на то, что в переменной `mPhoneIsSilent` правильно отображено состояние телефона и на его основе можно обновлять пользовательский интерфейс. Стратегическое решение разместить вызов `toggleUi()` в методе `onResume()` принято по той простой причине, что пользователь обычно запускает приложение Silent Mode Toggle, а затем возвращается к главному экрану и выключает телефон с помощью элементов интерфейса. Когда пользователь возвращается к деятельности, она возобновляется и выводится на передний план. В этот момент вызывается метод `onResume()`, который проверяет режим звонка и соответственно ему обновляет пользовательский интерфейс. Когда пользователь изменит режим, приложение отреагирует на это именно так, как ожидает пользователь, — изменит рисунок.

Установка приложения

Итак, вы создали свое первое приложение Android. В следующих нескольких разделах вы установите его в эмуляторе и физическом устройстве и запустите на выполнение.

Возвращаемся к эмулятору

Созданное вами приложение будет работать в эмуляторе (я уверен в этом, потому что сам пробовал). В предыдущей главе вы создали конфигурацию выполнения для запуска приложения Hello Android. Сейчас вы запустите приложение Silent Mode Toggle в этой же конфигурации выполнения. Надстройка ADT применит эту конфигурацию по умолчанию. Но сначала нужно установить приложение в эмуляторе. Для этого выполните следующие операции.

1. **В Eclipse выберите команду Run⇨Run (Выполнить⇨Выполнить) или нажмите клавиши <Ctrl+F11>, чтобы запустить приложение.**

   Активизируется диалоговое окно Run As (Выполнить как), показанное на рис. 5.3. Выделите пункт Android Application (Приложение Android) и щелкните на кнопке OK. Начнет устанавливаться эмулятор.

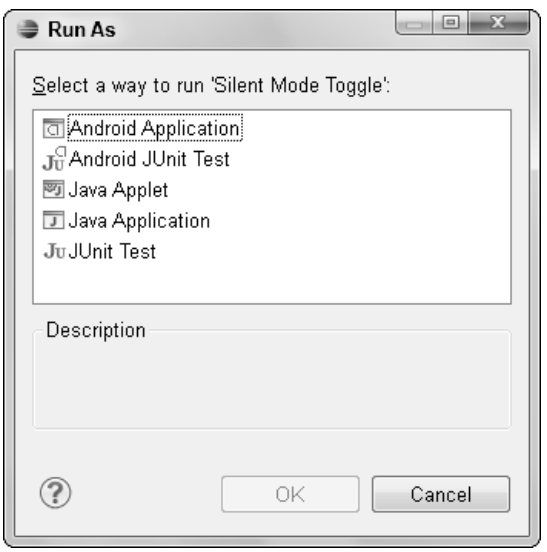


Рис. 5.3. Диалоговое окно, появляющееся при первом запуске приложения

2. **Подождите, пока будет установлен эмулятор (это займет около десяти минут) и разблокируйте его.**

   Если вы забыли, как разблокировать эмулятор, обратитесь к главе 3. Когда эмулятор разблокирован, приложение должно автоматически запуститься. Если этого не произошло, еще раз выберите команду Run⇨Run или нажмите клавиши <Ctrl+F11>. После этого приложение должно появиться в окне эмулятора (рис. 5.4).

Рис. 5.4. Приложение Silent Mode Toggle в эмуляторе

3. **Щелкните на кнопке Переключить режим звонка. Изображение изменится (рис. 5.5), что сигнализирует об успешном переключении режима.**
   Обратите внимание на новый значок (см. рис. 5.5), нарисованный операционной системой Android и сообщающий о бесшумном режиме звонка.

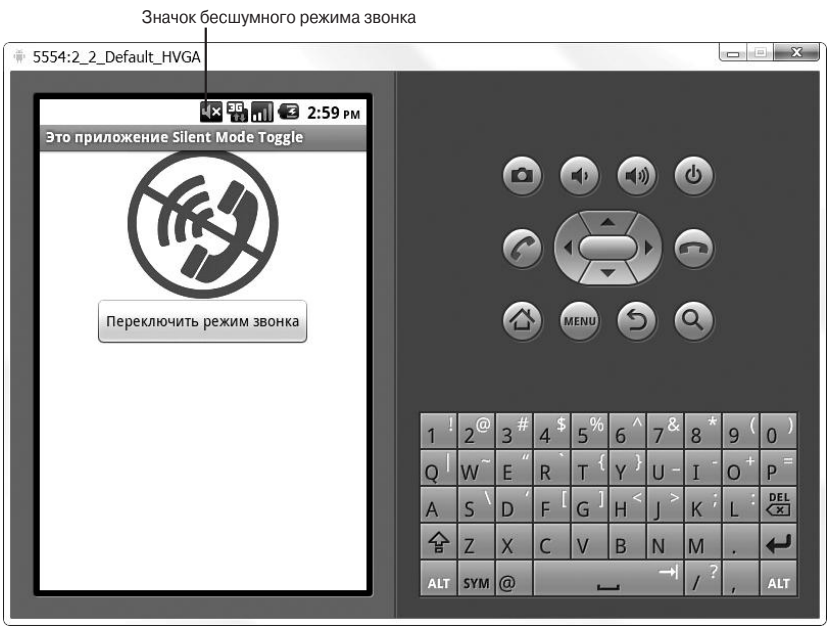


Рис. 5.5. Значок, сообщающий о бесшумном режиме

4. **Вернитесь в главное окно, щелкнув на правой панели эмулятора на кнопке главного окна (на ней нарисован домик).**

   Еще раз запустите приложение Silent Mode Toggle, чтобы поэкспериментировать с эмулятором. Для этого щелкните на средней кнопке в нижней части экрана. Активизируется список приложений, установленных в виртуальном устройстве. Щелкните на значке приложения Silent Mode Toggle, чтобы запустить его.

Установка приложения на физическое устройство Android

В среде Eclipse приложение устанавливается на физическое устройство точно так же, как в эмулятор. Но для этого нужно подготовить устройство, выполнив ряд операций. Если в главе 2 вы установили драйвер USB, подготовка устройства выполняется довольно просто (данный пример приведен для телефона Nexus One, в другом телефоне окна и команды могут быть другими).

1. **Включите установку приложений из источника, отличного от Android Market. Для этого выполните следующие операции.**
2. **На главном экране мобильного телефона выберите команду Menu⇨Settings (Меню⇨Параметры), чтобы открыть панель Settings. Выберите команду Applications (Приложения).**
3. **Установите флажок Unknown sources (Неизвестный источник), показанный на рис. 5.6.**

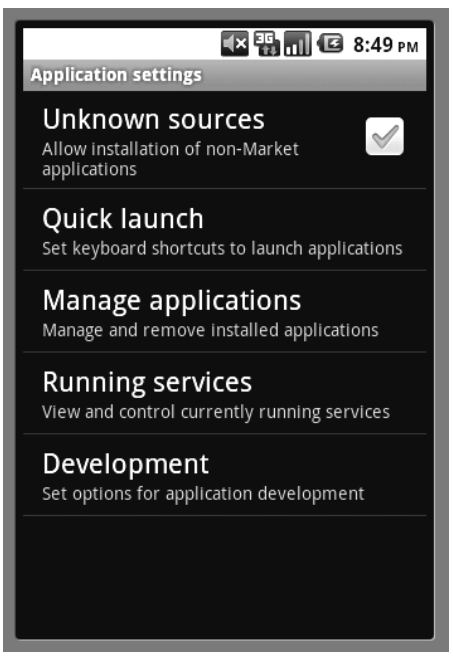


Рис. 5.6. Выбор разрешения устанавливать приложения с любого источника

Скорее всего, вы захотите не только запускать, но и отлаживать приложение на физическом устройстве.

4. **В окне Application settings (Параметры приложения) выберите команду Development (Разработка). В окне Development (рис. 5.7) установите флажок USB debugging (Отладка через USB).**

   Это позволит отлаживать приложение непосредственно на физическом устройстве (отладка рассматривается далее).

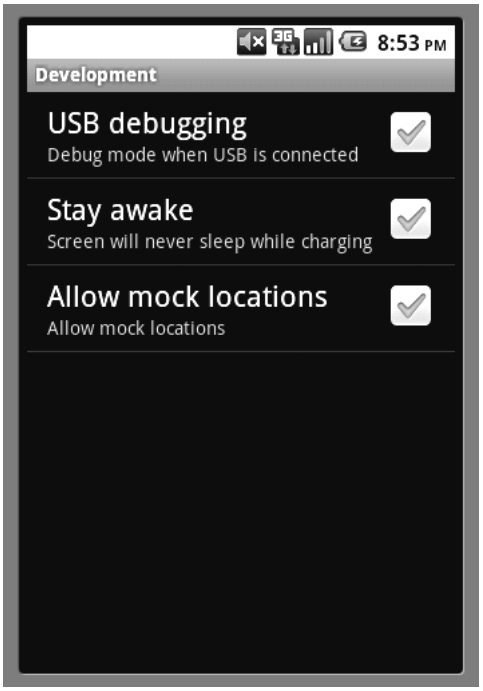


Рис. 5.7. Включение режима отладки через USB

5. **Присоедините мобильный телефон к компьютеру посредством кабеля USB.**
6. **Когда телефон будет обнаружен операционной системой компьютера, запустите приложение в Eclipse. Для этого выберите команду Run⇨Run (Выполнить⇨Выполнить) или нажмите клавиши <Ctrl+F11>.**

   В этот момент надстройка ADT распознает еще один вариант конфигурации выполнения и отображает диалоговое окно Android Device Chooser (Выбор устройства Android), в котором нужно выбрать устройство. На рис. 5.8 показано диалоговое окно Android Device Chooser, когда к компьютеру подключен мобильный телефон Nexus One. Его значок отличается от значка эмулятора, что позволяет отличить устройства друг от друга. Обратите внимание на то, что эмулятора не будет в списке, пока он не будет установлен и запущен на выполнение.
7. **Выберите в списке ваш мобильный телефон и щелкните на кнопке OK.**

   Приложение будет установлено в мобильном телефоне и запущено так же, как в эмуляторе. Через несколько секунд оно появится на экране мобильного телефона (значительно быстрее, чем в эмуляторе).

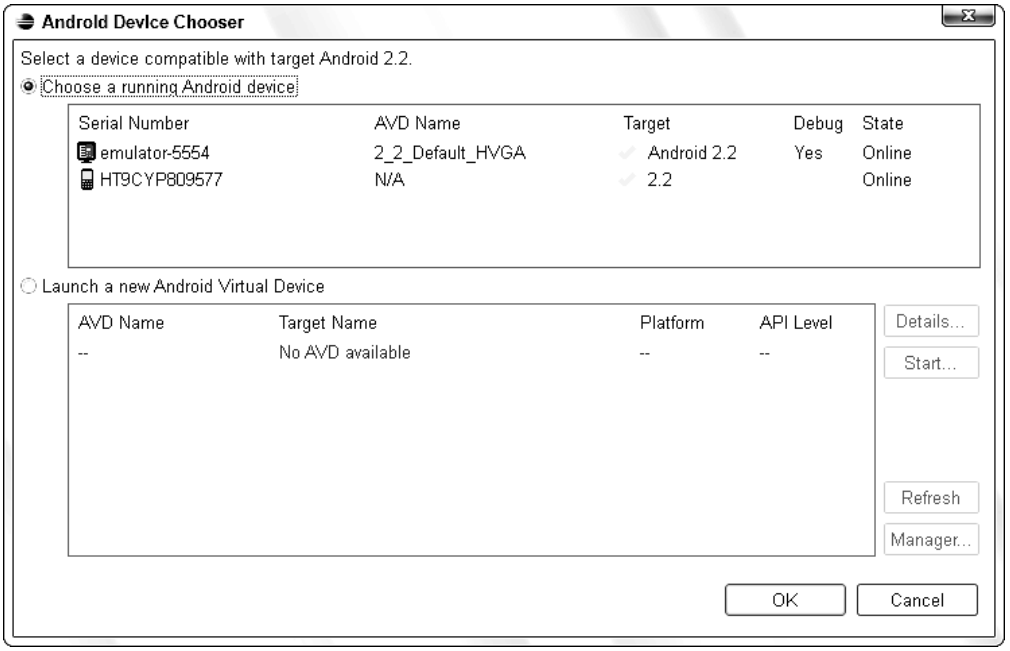


Рис. 5.8. Окно выбора устройства

Переустановка приложения

Как видите, установка приложения на физическое устройство — весьма простая задача. После установки флажков Unknown sources и USB debugging она решается так же, как и для эмулятора. Это справедливо и для повторной установки приложения. Чтобы переустановить приложение, не нужно делать что-либо особенное; все этапы этого процесса фактически рассмотрены выше. Приложение нуждается в переустановке после изменения кода в процессе отладки.

Состояние эмулятора

Программа Eclipse запускает эмулятор, но после этого эмулятор работает сам по себе и не зависит от Eclipse. Вы можете завершить Eclipse и, как ни в чем не бывало, продолжить работать с эмулятором и установленным в нем приложением Android.

Эмулятор и Eclipse "общаются" друг с другом посредством утилиты ADB (Android Debugging Bridge — мост отладки Android), которую вы установили вместе с надстройкой ADT.

Процесс переустановки

Чтобы переустановить приложение, достаточно выбрать в меню Eclipse команду Run⇨Run (Выполнить⇨Выполнить) или нажать клавиши <Ctrl+F11>. Вот и все! Надстройка ADT сама выяснит, изменилось ли приложение с момента последней установки и, следовательно, нужно ли перед запуском переустановить его в эмуляторе или физическом устройстве.

Отладка

Предположим, вы написали прекрасный код, который немедленно заработал без всякой отладки. Однако, должен вам признаться, мне это никогда не удавалось сделать. Да и вам вряд ли когда-нибудь удастся. Даже не пытайтесь. Настоящий профессионализм состоит не в том, чтобы написать код без ошибки, а в том, чтобы уметь отладить приложение. Когда приложение работает не так, как планировалось, необходимо выяснить, почему. Это очень сложная задача. Не менее сложная, чем разработка приложения. Чтобы облегчить ее решение, надстройка ADT предоставляет ряд ценных инструментов, позволяющих отслеживать работу приложения и вылавливать ошибки.

Инструмент DDMS

Инструмент отладки DDMS (Dalvik Debug Monitor Server — сервер мониторинга и отладки в виртуальной машине Dalvik) предоставляет ряд полезных средств отладки (список неполный):

- ✓ перенаправление портов;
- ✓ перехват экрана;
- ✓ извлечение информации о потоках и кучах устройства;
- ✓ предоставление дампа системных сообщений;
- ✓ информация о состояниях процессов и радиоприемников;
- ✓ спуфинг входных звонков и SMS;
- ✓ спуфинг географических данных.

Инструмент DDMS может работать как с эмулятором, так и с подключенным физическим устройством. Он находится в папке `tools` пакета Android SDK. В главе 1 вы добавили маршрут папки `tools` в список системных маршрутов доступа, поэтому сейчас можете использовать DDMS как утилиту командной строки.

Что нужно знать о DDMS

Отладка — всегда тяжелая работа. В счастью, DDMS значительно облегчает ее, предоставляя ряд полезных средств. Одно из наиболее часто используемых средств DDMS — диалоговое окно LogCat, которое позволяет просматривать журнал системных сообщений Android (рис. 5.9). В журнале вы увидите базовые информационные сообщения, включая сведения о состоянии приложения и устройства, а также предупреждения и сообщения об ошибках. Во время работы приложения вы можете получить сообщение Application Not Responding (Приложение не отвечает) или Force Close (Принудительное завершение), из которого не видно, что произошло. В таком случае запустите DDMS и просмотрите записи в окне LogCat, в котором вы сможете идентифицировать, где произошло исключение, вплоть до номера строки. Инструмент DDMS не решит проблему вместо вас, но он поможет вам отследить причину ошибки и облегчит ее поиск и устранение.

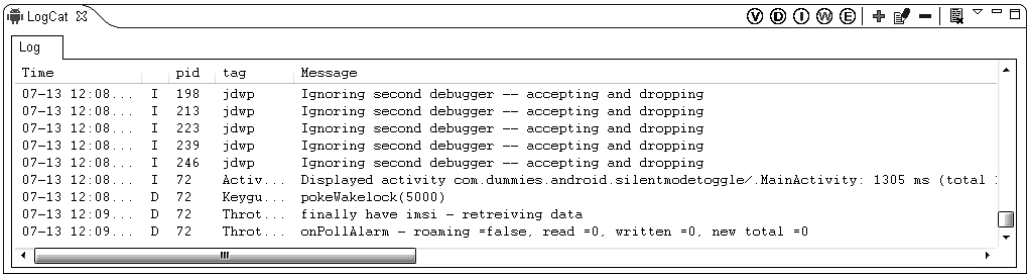


Рис. 5.9. Журнал системных сообщений

Инструмент DDMS очень полезен также в ситуации, когда у вас нет физического устройства, на котором можно протестировать приложение. Характерный пример такой ситуации — разработка приложения, в котором GPS и Google MapView используются для вывода карты на экран. Изображение на карте зависит от того, где и в какое время суток вы находитесь. Следовательно, чтобы отладить и протестировать приложение, придется ездить с ним по всему миру или, как минимум, по своей стране. Конечно, такой режим отладки нереален. К счастью, отладить подобное приложение вам поможет инструмент DDMS, который предоставляет средства управления местоположением. Как разработчик, вы можете вручную задать координаты GPS или создать файл в формате GPX или KML, представляющий время и точку на карте. Можете даже задать перемещение, например оставаться здесь в течение пяти минут, после чего начать движение по данной дороге на запад со скоростью 90 км/ч.

Как вставить свои сообщения в журнал DDMS

Чтобы вставить в код Java сообщение, отображаемое в окне DDMS, достаточно одной строки кода. Откройте файл `MainActivity.java` и в конце метода `onCreate()` введите запись журнала, как показано в листинге 5.5.

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   mAudioManager =
            (AudioManager)getSystemService(AUDIO_SERVICE);
   checkIfPhoneIsSilent();
   setButtonClickListener();
   Log.d("SilentModeApp", "Это моя запись");                    12
}
```

Код, приведенный в строке 12, демонстрирует вставку сообщения в системный журнал. Строка `SilentModeApp` называется *дескриптором*, или параметром `TAG`. Это имя, присваиваемое записи в журнале. Вторая строка — это текст сообщения, отображаемого в окне DDMS. Параметр `TAG` позволяет фильтровать сообщения при их просмотре в окне DDMS.

Рекомендуется объявить в коде Java константу `TAG` и применять ее вместо многократного ввода дескриптора сообщения.

```
private static final String TAG = "SilentModeApp"
```

Обратите внимание на имя поля `Log.d` в листинге 5.5. Имя `d` означает отладочное сообщение. Доступны также следующие типы сообщений:

- ✓ **`e`** — ошибка;
- ✓ **`I`** — информация;
- ✓ **`wtf`** — “What The...” (Что за ерунда?!);
- ✓ **`v`** — “Verbose” (Подробная информация).

Типы сообщений используются, чтобы задать, как они должны записываться в журнал и обрабатываться в процессе отладки приложения.

Чтобы журнал работал, нужно импортировать пакет `android.util.Log`.

Просмотр сообщений DDMS

Чтобы просматривать сообщения DDMS, нужно либо запустить утилиту DDMS вручную, либо открыть окно DDMS в рабочей среде Eclipse.

- ✓ **Вручную.** Откройте папку, в которой вы установили Android SDK. Откройте в ней папку `tools` и дважды щелкните на файле `ddms.bat`. Утилита DDMS будет запущена вне рабочей среды Eclipse и независимо от нее (рис. 5.10).

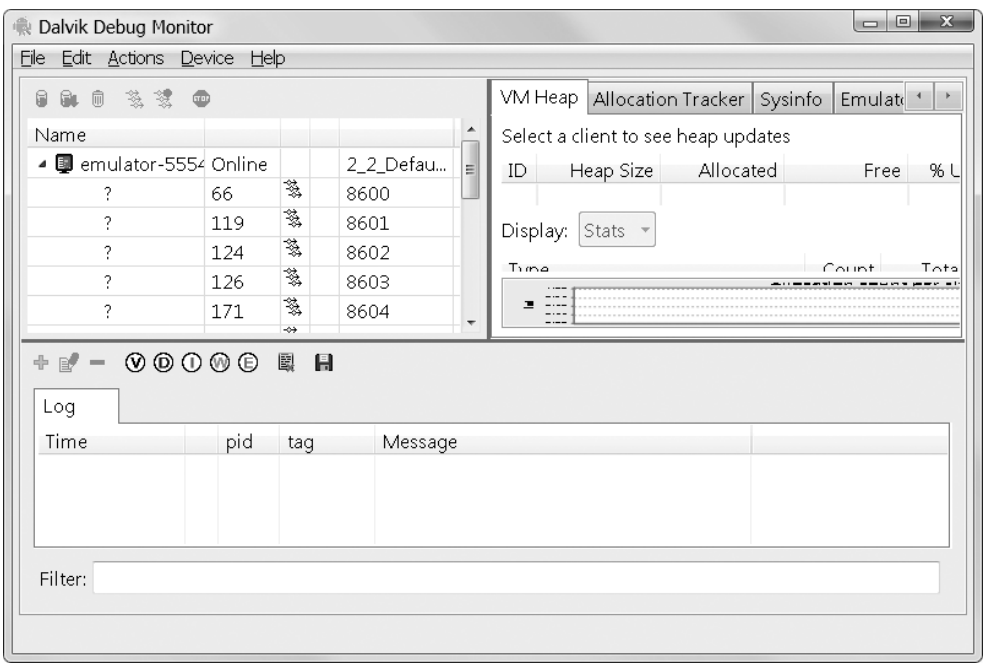


Рис. 5.10. Экземпляр DDMS, выполняющийся отдельно от Eclipse

✓ **В Eclipse.** Надстройка ADT уже установила инструмент DDMS в рабочую среду Eclipse. Чтобы открыть окно DDMS, щелкните в рабочей среде Eclipse на кнопке Open Perspective (Открыть окно), показанной на рис. 5.11, и в выберите в открывшемся меню команду DDMS. Если в меню нет пункта DDMS, выберите команду Other (Другие) и выберите DDMS. В рабочей среде Eclipse в список окон будет добавлено окно DDMS, которое после этого легко можно будет открыть. В частности, после выбора команды DDMS соответствующее окно открывается автоматически. В окне DDMS есть вкладка LogCat. Обычно она расположена в нижней части экрана. Я предпочитаю переместить ее в главную область экрана (рис. 5.12). Для этого перетащите корешок вкладки LogCat вправо вверх и отпустите кнопку мыши, когда черный прямоугольник займет нужное место.

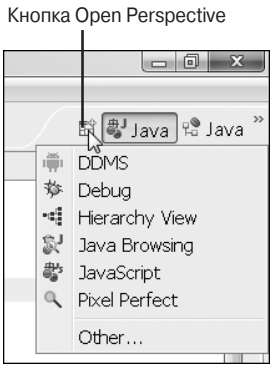


Рис. 5.11. Открытие окна DDMS

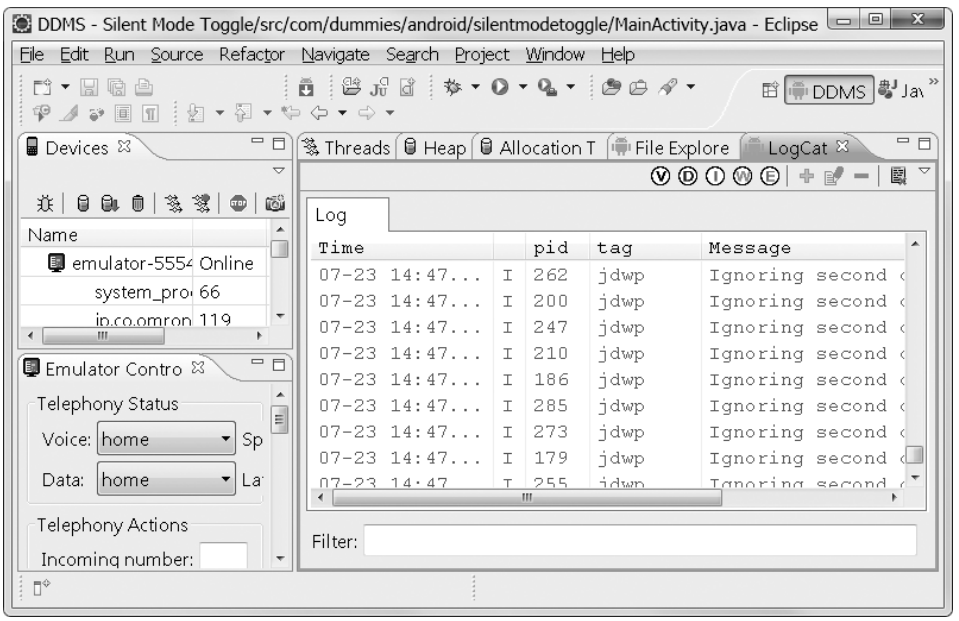


Рис. 5.12. Вкладка LogCat в рабочей среде Eclipse

Теперь в рабочей среде Eclipse запустите приложение, выбрав команду Run⇨Run или нажав клавиши <Ctrl+F11>. Когда приложение появится в эмуляторе, откройте окно DDMS. Во вкладке LogCat вы увидите результат выполнения инструкции `Log.d` (см. выше), которая вставила запись в журнал (рис. 5.13). Другие системные сообщения могут отличаться от приведенных в книге, но ваша запись будет точно такой, какой вы ее ввели в инструкции `Log.d`.

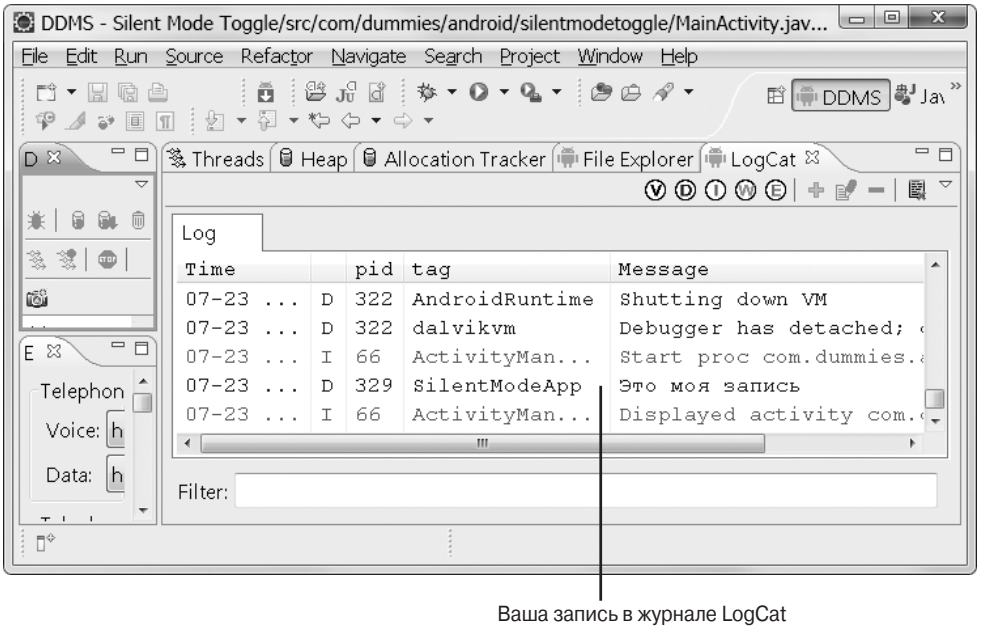


Рис. 5.13. В журнале видна запись, вставленная инструкцией `Log.d`

Теперь можете переключиться обратно в окно Java. Для этого щелкните на кнопке Java Perspective (Окно Java), показанной на рис. 5.14.

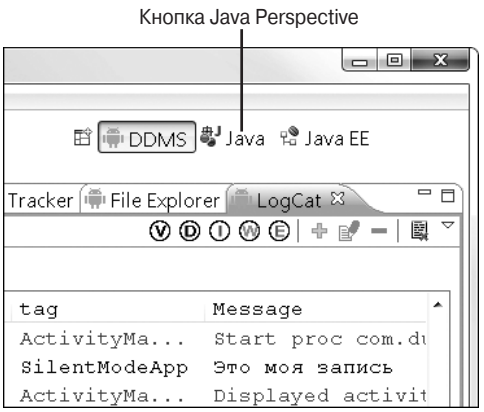


Рис. 5.14. Переключение обратно в окно редактора Java

Использование отладчика Eclipse

Утилита DDMS чрезвычайно полезна, но все же ваше главное оружие — отладчик, встроенный в рабочую среду Eclipse. С его помощью вы сможете устанавливать точки прерывания, проверять значения переменных в окне наблюдений, просматривать журнал LogCat и выполнять многие другие операции.

Отладчик полезен для вылавливания ошибок любого типа, как логических, так и времени выполнения. Впрочем, синтаксические ошибки (наиболее легкие) в отладчик не попадают, потому что компилятор Eclipse перехватывает их. Если в коде есть синтаксическая ошибка, приложение не компилируется, и рабочая среда Eclipse сообщает вам об этом. Причем не просто сообщает об этом факте, но и показывает, в каком месте совершена ошибка, подчеркнув проблемную инструкцию красной волнистой линией. Правда, если есть ошибка, компилятор может не понять, что вы хотели сделать, и подчеркнуть другую, соседнюю инструкцию, но, взглянув на соседние инструкции, вы без труда поймете, в чем дело.

Ошибки времени выполнения

Это довольно неприятный тип ошибок. Они, как птица Феникс, возникают из ниоткуда и оставляют после себя полный хаос. В Android ошибки выполнения происходят, как нетрудно догадаться, во время выполнения приложения. Чаще всего ошибка происходит (или, по крайней мере, проявляется) в момент, когда пользователь что-либо сделал, например нажал кнопку или выбрал пункт меню. Причины ошибки могут быть практически любые. Возможно, вы не инициализировали экземпляр `AudioManager` в методе `onCreate()` или отложили его инициализацию на более поздний момент, решив, что пользователь пока что не будет включать звук. Затем приложение обратилось к переменной `mAudioManager`, когда она еще не инициализирована, и операционная система сгенерировала исключение времени выполнения.

Отладчик поможет вам найти такую ошибку. Например, вы можете установить точку прерывания в конце метода `onCreate()` и просмотреть значения переменных в окне отладки. Увидев, что переменная `mAudioManager` не инициализирована, вы сразу поймете, в чем дело. В листинге 5.6 данная ситуация создана искусственно. Инструкция инициализации `mAudioManager` закомментирована, в результате чего во время выполнения будет сгенерировано исключение.

Листинг 5.6. Инициализация переменной `mAudioManager` закомментирована

```
private AudioManager mAudioManager;                          1
private boolean mPhoneIsSilent;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  //mAudioManager =                                          9
  // (AudioManager)getSystemService(AUDIO_SERVICE);
  checkIfPhoneIsSilent();
  setButtonClickListener();
  Log.d("SilentModeApp", "Это тестовое приложение");
}

/**
* Выяснение режима звонка
*/
private void checkIfPhoneIsSilent() {
```

```
  int ringerMode = mAudioManager.getRingerMode();          22
  if (ringerMode == AudioManager.RINGER_MODE_SILENT) {
    mPhoneIsSilent = true;
  } else {
    mPhoneIsSilent = false;
  }
}
```

Ниже приведено описание соответствующих строк кода.

- ✓ **1.** Объявление переменной `mAudioManager` на уровне класса.
- ✓ **9.** Предположим, что для тестирования звука я закомментировал этот код, а затем "забыл" удалить символы комментариев.
- ✓ **22.** Когда метод `onCreate()` вызывает метод `checkIfPhoneIsSilent()`, приложение создает исключение времени выполнения, потому что переменная `mAudioManager` равна `null`. Тем не менее код попытался сослаться на этот объект (хотя он не существует).

Таким образом, применение отладчика для проверки метода `onCreate()` позволяет отследить причину возникновения ошибки.

Создание точек прерывания

Существует несколько способов создания точки прерывания.

- ✓ В окне редактора Java щелкните на строке кода, чтобы выделить ее. Выберите команду Run⇨Toggle Breakpoint (Выполнить⇨Переключить точку прерывания), как показано на рис. 5.15.
- ✓ С помощью мыши выделите строку, в которую нужно установить точку прерывания, и нажмите клавиши <Ctrl+Shift+B>. Эту комбинацию клавиш можно также увидеть в меню на рис. 5.15.
- ✓ Дважды щелкните на серой полоске слева от кода в редакторе Eclipse напротив строки, в которой нужно создать точку прерывания.

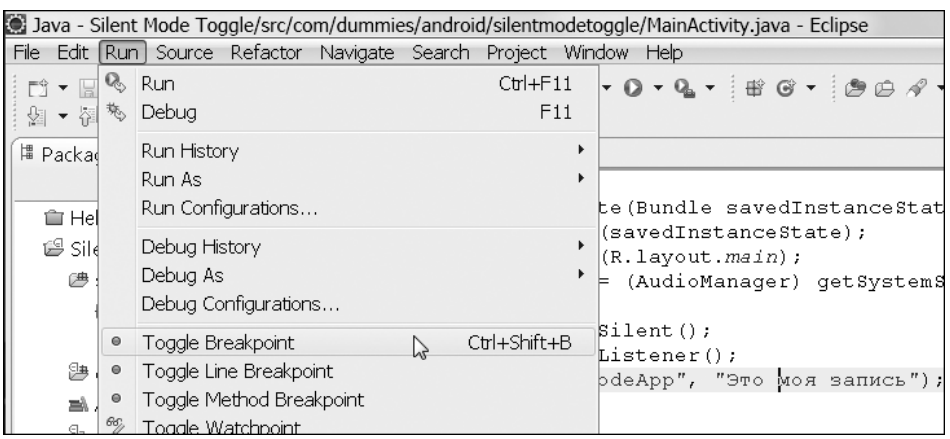


Рис. 5.15. Установка точки прерывания с помощью меню или горячих клавиш

При использовании любого из этих методов на серой полоске слева от кода появляется маленький круглый значок (рис. 5.16), сигнализирующий о наличии точки прерывания.

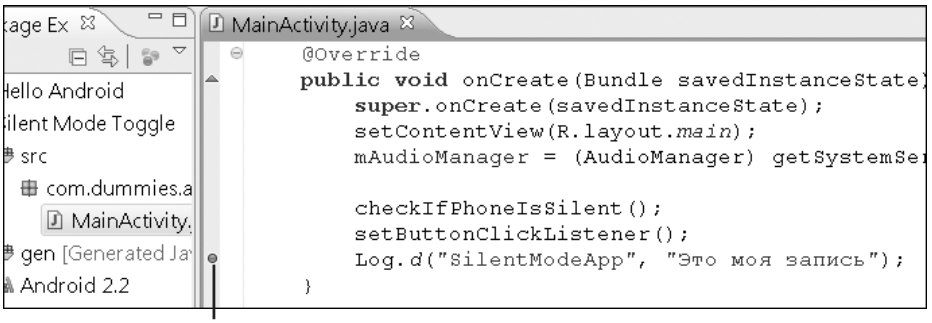


Рис. 5.16. Точка прерывания в окне редактора Java

Давайте поэкспериментируем с отладчиком. Закомментируйте строку 3 в методе `onCreate()`, как показано в листинге 5.7.

Листинг 5.7. Искусственное создание ошибки

```
setContentView(R.layout.main);

//mAudioManager =
//     (AudioManager)getSystemService(AUDIO_SERVICE);        3

checkIfPhoneIsSilent();                                     5
```

- ✓ **3.** Отмена инициализации переменной `mAudioManager`.
- ✓ **5.** Этот вызов метода приведет к краху приложения.

Установите точку прерывания в строке 5.

Работа с отладчиком

Чтобы приложение было доступным для отладчика, нужно переключить его в режим отладки, т.е. отметить как отлаживаемое. Для этого откройте файл `AndroidManifest.xml`, дважды щелкнув на его имени в окне Package Explorer (Обозреватель пакетов), расположенном на левой панели Eclipse. Откройте вкладку Application (Приложение). В раскрывающемся списке Debuggable (Доступный для отладки) выберите значение true (рис. 5.17). Сохраните файл `AndroidManifest.xml`.

Если не переключить приложение в режим отладки, его нельзя будет отлаживать, потому что оно не будет подключено к отладчику. Я сам часто забываю выбрать в раскрывающемся списке Debuggable значение true. К счастью, рабочая среда Eclipse настойчиво напоминает об этом тем, что не предоставляет инструменты отладки. В таком случае найдите этот раскрывающийся список и установите значение true.

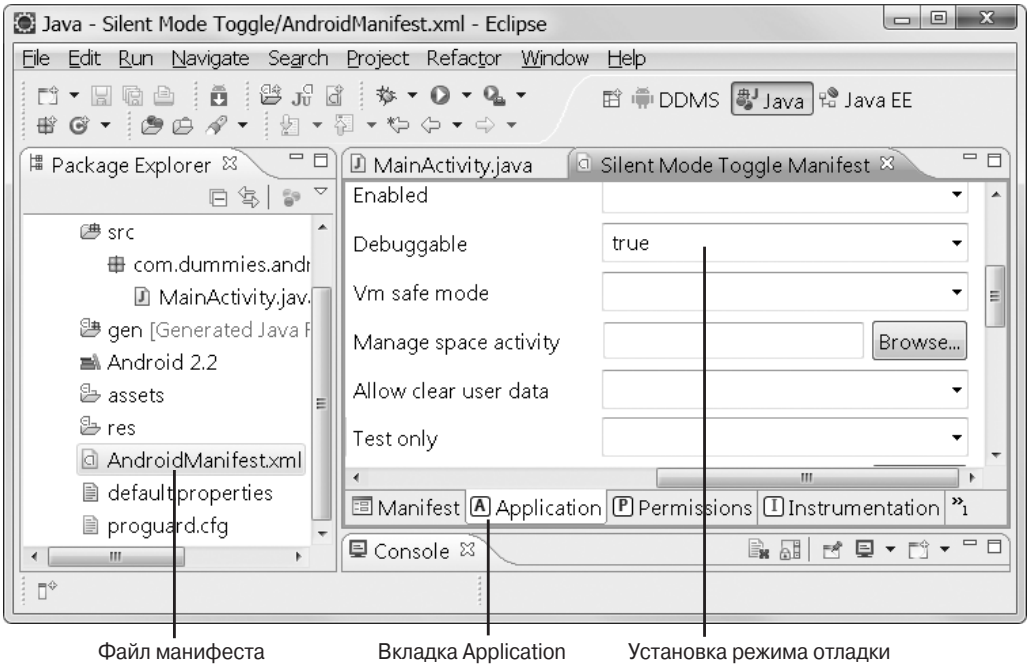


Рис. 5.17. Переключение приложения в режим отладки

Итак, вы создали код с умышленной ошибкой (см. листинг 5.7) и готовы приступить к его отладке.

Чтобы запустить отладчик, выберите команду Run⇨Debug (Выполнить⇨Отладка) или нажмите клавишу <F11>. Этим вы приказываете надстройке ADT и программе Eclipse установить приложение в эмулятор или физическое устройство и подключить к приложению отладчик.

Если эмулятор не выведен на передний план, сделайте это. Приложение будет установлено, и в эмуляторе появится диалоговое окно, сообщающее о том, что ADT и эмулятор подключают отладчик (рис. 5.18).

Немного подождите. Не щелкайте на кнопке Force Close (Принудительное закрытие). Через несколько секунд отладчик будет подключен, и код приложения начнет выполняться в эмуляторе. Когда приложение дойдет до точки прерывания, выполнение будет остановлено и появится диалоговое окно (рис. 5.19), спрашивающее, нужно ли в рабочей среде Eclipse открыть окно Debug (Отладка). Щелкните на кнопке Yes.

Теперь приложение остановлено в точке прерывания (рис. 5.20), и можно исследовать его с помощью отладчика в интересующий вас момент выполнения. Вы как бы приказали: “Остановись, мгновенье!” Сейчас можно, например, навести указатель на любую переменную и увидеть ее текущее значение.

Рис. 5.18. Эмулятор ждет подключения отладчика

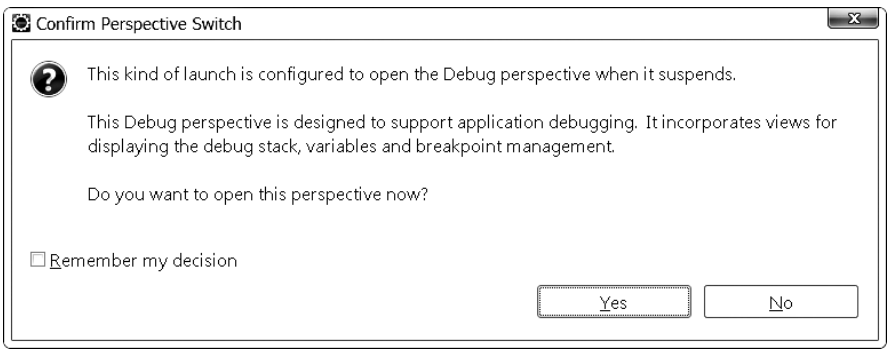


Рис. 5.19. Открытие окна Debug

Наведите указатель на переменную `mAudioManager`, и вы увидите, что она равна `null`, т.е. не инициализирована. Естественно, ведь вы закомментировали инструкцию ее инициализации.

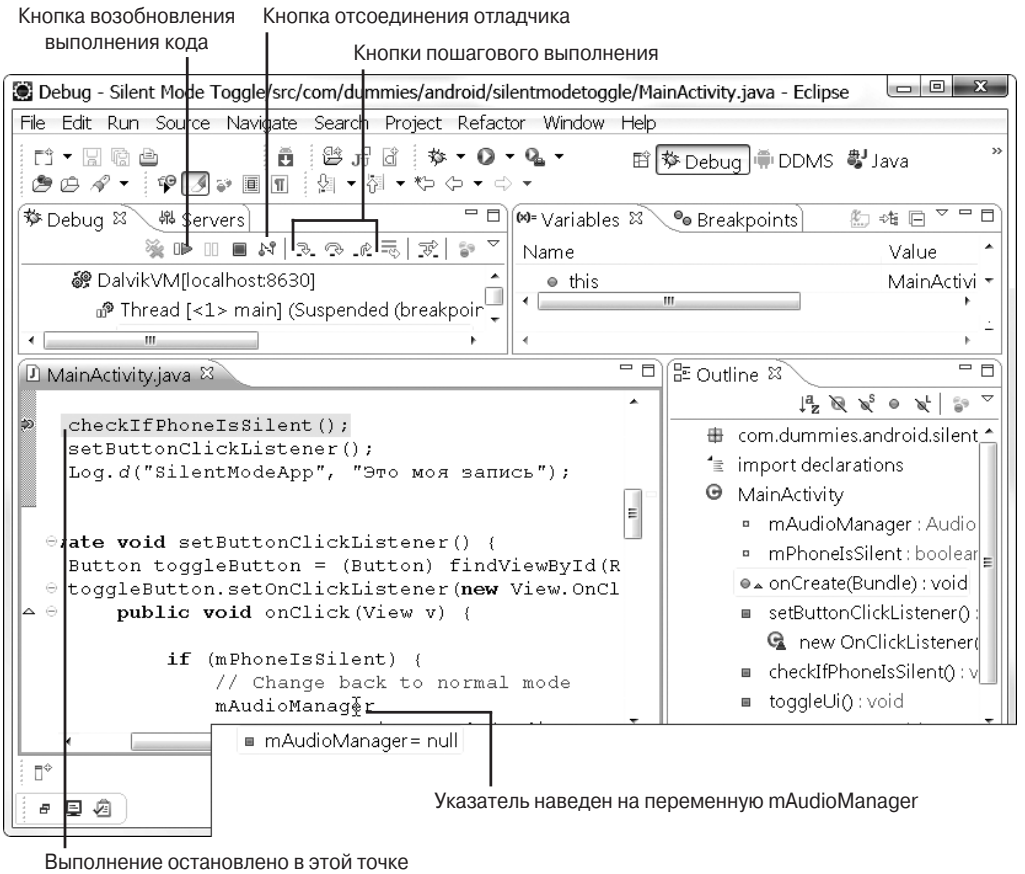


Рис. 5.20. Код приложения в окне отладчика

С помощью кнопок пошагового выполнения можно остановить приложение на любой инструкции, чтобы посмотреть, чему равны переменные в этой точке. Если три раза щелкнуть на кнопке Resume (Возобновить) или нажать клавишу <F8>, окно отладки изменится и появится сообщение `source not found` (источник не найден). Откройте окно эмулятора, и вы увидите, что в нем появилось сообщение о крахе приложения (рис. 5.21). Щелкните на кнопке Force close (Принудительное закрытие). Операционная система выводит данное окно в эмуляторе или устройстве, если сгенерировано исключение времени выполнения, не обработанное в коде приложения. Вы как разработчик должны не допустить появления этого окна при работе своего приложения.

Чтобы отключить отладчик, щелкните на кнопке Disconnect (Отсоединить), показанной на рис. 5.20. Вернитесь в окно редактора Java и снимите символы комментария в строке 3 (см. листинг 5.7) файла `MainActivity.java`. Вы ведь не хотите оставить в приложении ошибку, с которой экспериментировали в окне отладки.

Рис. 5.21. Диалоговое окно принудительного закрытия приложения вследствие ошибки времени выполнения

Логические ошибки

Любой компьютер делает только то, что программист заложил в него. Это же справедливо и для устройства Android. Ни устройство, ни приложение не знают, что должно быть сделано; они делают лишь то, что вы им приказали. Они никогда не ошибаются, ошибиться можете только вы. К счастью, большинство ошибок вылавливает компилятор, но те, которые остаются (они называются *логическими ошибками*), — самые трудные. Пример логической ошибки приведен в листинге 5.8.

Листинг 5.8. Код не всегда правильно идентифицирует режим звонка

```
/**
  * Переключение изображения при изменении режима звонка
  * с громкого на бесшумный или наоборот
  */

private void toggleUi() {
  ImageView imageView =
    (ImageView) findViewById(R.id.phone_icon);
      Drawable newPhoneImage;

  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();
  };
```

- ✓ **11.** Оператор `if` проверяет, работает ли телефон в бесшумном режиме. Об этом сигнализирует значение булевой переменной `mPhoneIsSilent`. Если оно равно `true`, телефон находится в бесшумном режиме, а при значении `false` включен режим громкого звонка.
- ✓ **26.** Чтобы метод `toggleUi()` отобразил правильный рисунок в пользовательском интерфейсе, приложение должно знать текущий режим звонка. В строке 26 я "случайно" закомментировал вызов метода `checkIfPhoneIsSilent()`, который обновляет переменную `mPhoneIsSilent` на уровне класса. Поскольку обновления не происходит в методе `Resume()`, возможна следующая ситуация. Пользователь выходит из приложения Silent Mode Toggle, изменяет режим звонка с помощью параметров мобильного телефона и возвращается в приложение Silent Mode Toggle. Выполняется метод `Resume()`, но значение `mPhoneIsSilent` не изменяется, т.е. теперь оно неправильное. С помощью отладчика можно установить точку прерывания в первой строке метода `toggleUi()` и проверить значения разных переменных. В этот момент вы увидите, что переменная `mPhoneIsSilent` имеет значение `null`, из чего нетрудно понять, что это произошло вследствие логической ошибки.

Выход за границы приложения

Иногда устройство решает задачи, внешние по отношению к данному приложению, но тем не менее влияющие на него. Например, устройство может загружать большой файл в фоновом режиме во время прослушивания музыки с помощью онлайн-радио. Нарушит ли процесс загрузки файла из Интернета работу приложения, воспроизводящего радиопередачу? Ответ на этот вопрос зависит от ряда факторов. Еще один вопрос: если приложению необходимо соединение с Интернетом и по какой-либо причине оно не может получить его, потерпит ли оно крах? Как оно отреагирует на невозможность получения доступа к Интернету? Эти и другие аналогичные вопросы я называю "выходом за границы приложения".

Не все приложения одинаково качественные. Я видел много хороших приложений, но встречал также немало плохих. Перед окончательной сборкой и поставкой

своего первого приложения Android убедитесь в том, что вы знаете все его тонкости и учли все, что может повлиять на его работу. Вы должны гарантировать, что оно не потерпит крах, когда пользователь выполняет с мобильным устройством рутинные операции, не связанные с работой приложения.

Проектирование и создание приложения для мобильного устройства существенно отличается от решения этой же задачи для настольного компьютера. Причина этого простая: ресурсов у мобильного устройства (объем памяти, мощность процессора, разрешение экрана и т.п.) существенно меньше, чем у настольного. Если устройством Android является мобильный телефон, его главная задача — решение "телефонных" задач, таких как прием входящих звонков и SMS, включение сигнала, поддержка телефонной книги и пр.

Когда пользователь говорит по телефону, операционная система Android считает этот процесс главным. Если в это же время происходит загрузка файла в фоновом режиме, этот процесс считается второстепенным. Если более приоритетному процессу не хватает ресурсов, Android уничтожает менее приоритетные процессы. Файл всегда можно загрузить повторно, но если не ответить на телефонный звонок, последствия могут быть непоправимыми. Поэтому вы обязательно должны протестировать ситуацию, в которой приложение загрузки файла уничтожается в непредсказуемый момент времени по независящим от него причинам. Также протестируйте ситуации, в которых сигнал беспроводного канала связи становится слабым или пропадает. Соединение будет отключено и файл не загрузится. Продолжит ли приложение загрузку автоматически при восстановлении соединения? Протестируйте все возможные варианты развития событий и обеспечьте правильную работу приложения в любой ситуации. В противном случае ваше приложение будет генерировать исключения времени выполнения, пользователи будут недовольны, и вы получите отрицательные отзывы на сайте Android Market.

Взаимодействие с приложением

Дабы убедиться в том, что приложение работоспособно, запустите его и поработайте с ним. Когда ваше приложение выполняется, запустите другое приложение, например браузер. Немного погуляйте по Интернету и вернитесь к вашему приложению. Нажмите в нем несколько кнопок и посмотрите, что происходит. Не изменилось ли что-нибудь? Поэкспериментируйте с разными средствами приложения. Проверьте, все ли они работают так, как ожидалось. Проверьте, что произойдет, если во время работы приложения звонит телефон и пользователь отвечает на звонок. Сохраняется ли состояние приложения в методе `onPause()` и восстанавливается ли оно в методе `onResume()`? Операционная система Android позаботилась об автоматическом решении многих задач, но в конечном счете за правильную работу своего приложения отвечаете вы.

Тестирование приложения

Запустите в эмуляторе приложение Silent Mode Toggle, щелкнув на его значке в списке приложений. На данный момент вы уже выполнили первый этап тестирования: убедились в том, что приложение запускается.

Когда приложение открыто, проверьте, включен ли бесшумный режим звонка, о котором сигнализирует значок в строке извещений (рис. 5.22).

Рис. 5.22. Когда телефонная трубка зачеркнута, в строке извещений должен быть значок бесшумного режима

Щелкните на кнопке Переключить режим звонка. Изменяется ли изображение? Поэкспериментируйте с приложением, дабы убедиться в том, что оно работает, как от него ожидается. Если найдете какой-нибудь дефект, выясните его причину с помощью инструментов отладки, рассмотренных ранее.

Автоматическое тестирование

В последнее десятилетие все более популярными становятся технологии гибкой разработки и автоматического тестирования. На платформе Android они еще не так популярны, но это вопрос ближайшего будущего. В составе пакета Android SDK вы инсталлировали инструменты модульного тестирования, которые можно использовать не только для классов Java, но и базовых классов Android, включая интерактивные элементы интерфейса. Информация о модульном тестировании приведена в документации Android по адресу `http://d.android.com/guide/topics/testing/testing_android.html`.

О модульном тестировании приложений Android можно написать целую книгу объемом значительно большим, чем эта, поэтому я лишь кратко упомяну о двух наиболее интересных инструментах автоматического тестирования.

- ✓ **jUnit.** При установке SDK выполняется интеграция платформы jUnit с надстройкой ADT. jUnit — это весьма популярная платформа модульного тестирования, применяемая в Java. Инструменты jUnit можно использовать как для модульного тестирования, так и для тестирования интерактивных средств Android. Дополнительную информацию о jUnit

можно найти по адресу `www.junit.org`. В Eclipse встроены инструменты тестирования, облегчающие использование средств jUnit.

- ✓ **Monkey.** Это программа проверки пользовательских интерфейсов приложений, выполняющаяся в эмуляторах и физических устройствах и генерирующая потоки псевдослучайных пользовательских событий. Автоматически генерируются такие события, как прикосновения к экрану, жесты, щелчки, телефонные звонки и все системные события. Программа Monkey — хороший инструмент стресс-тестирования приложения. Она устанавливается вместе с Android SDK.

Глава 6

Ресурсы Android

В этой главе...

- Понятие ресурсов
- Работа с ресурсами
- Глобализация приложения с помощью ресурсов

В предыдущих главах мы уже неоднократно рассматривали ресурсы Android. Зачем же возвращаться к ним опять и посвящать им отдельную главу? Информация о ресурсах и их использовании, приведенная в главах 3-4, была необходима для понимания назначения папки ресурсов и способов их использования в простом приложении для его компоновки. Однако есть много других применений ресурсов Android в приложении, включая задачу глобализации приложения, которую мы рассмотрим в данной главе.

Типы ресурсов

В Android поддерживаются следующие типы ресурсов:

- ✓ компоновки;
- ✓ строки;
- ✓ изображения;
- ✓ размеры;
- ✓ стили;
- ✓ темы;
- ✓ значения;
- ✓ меню;
- ✓ цвета.

Вы уже знакомы с изображениями, компоновками и строками, потому что эти ресурсы применялись в приложении Silent Mode Toggle, которое служило примером в предыдущих главах. Это наиболее распространенные типы ресурсов, и они используются практически в каждом приложении Android. В следующих разделах мы рассмотрим другие типы ресурсов.

Размеры

В ресурсах Android *размер* — это число, после которого приведена единица измерения, например `10px`, `2in`, `5sp`. Размеры используются при задании в Android любых

свойств, для которых кроме числа необходима единица измерения. Например, ширину внутренней пустой полоски (padding) элемента интерфейса можно задать как `10px`, т.е. десять пикселей. Операционная система Android поддерживает следующие единицы измерения.

- ✓ **`dp` (density-independent pixels — пиксели, независимые от разрешения).** Эта абстрактная единица измерения основана на физической плотности точек на экране относительно разрешающей способности 160dpi (dots per inch — точек на дюйм). Значение `1dp` эквивалентно одному пикселю на экране с разрешающей способностью 160dpi. Отношение `dp` к пикселям зависит от плотности точек, но не обязательно пропорционально. При компоновке интерфейса эта единица измерения используется чаще других. Вопросы правильного выбора значений `dp` довольно сложны, но их нужно решить, если вы планируете активно разрабатывать приложения для экранов с разным разрешением. Дополнительную информацию о поддержке экранов с разным разрешением можно найти по такому адресу:

  `http://developer.android.com/guide/practices/screens_support.html`
- ✓ **`sp` (scale-independent pixels — пиксели, независимые от масштаба).** Эта единица измерения аналогична единице `dp`, но масштабируется соответственно предпочтениям, установленным пользователем для размеров шрифта. Единицу измерения `sp` рекомендуется использовать, если в приложении задаются размеры шрифта.
- ✓ **`pt` (points — пункты).** Один пункт равен 1/72 дюйма.
- ✓ **`px` (pixels — пиксели).** Фактическое количество пикселей на экране. На первый взгляд, это самая простая и беспроблемная единица, но использовать ее не рекомендуется. Приложение, разработанное с использованием единиц `px`, прекрасно выглядит на экране с разрешением, для которого оно создавалось, однако при другом разрешении экрана элементы интерфейса искажаются, надписи становятся неразборчивыми, рисунки обрезаются и происходят многие другие неприятности.
- ✓ **`mm` (миллиметры).** Размер непосредственно на экране в миллиметрах.
- ✓ **`in` (inches — дюймы).** Один дюйм равен 2,54 сантиметрам.

Стили

Стили, как несложно догадаться, предназначены для стилизации внешнего вида приложения. В Android они похожи на стили CSS (Cascading Style Sheets — каскадные таблицы стилей), используемые в веб-дизайне. *Стиль* — это коллекция свойств, примененная к некоторому представлению (в файле компоновки), деятельности или всему приложению (в файле манифеста). Стили Android поддерживают наследование. Это означает, что можно определить базовый стиль, после чего создавать на его основе другие стили и при необходимости редактировать стиль в каждом конкретном случае применения в приложении. К свойствам стилей относятся размер шрифта, цвет шрифта, цвет фона и т.п.

Темы

Тема — это стиль, примененный ко всей деятельности или приложению, а не к индивидуальному представлению. Когда стиль применен как тема, каждое представление приложения или деятельности наследует его свойства. Например, можно присвоить всем представлениям `TextView` определенный шрифт и установить этот шрифт в теме деятельности или приложения. Тогда во всех представлениях деятельности или приложения текст будет отображаться этим шрифтом.

Значения

Ресурс значений может содержать различные типы данных, включая следующие.

- ✓ **Булево значение.** Определено в формате XML и хранится в файле `/res/values/<имя_файла>.xml`. Имя файла произвольное, например `bools.xml`.
- ✓ **Целочисленное значение.** Тоже определено в формате XML и хранится в файле `/res/values/<имя_файла>.xml`. Имя файла может быть произвольным, например `integers.xml`.
- ✓ **Целочисленный массив.** Массив целых чисел, хранящийся в формате XML в файле `/res/values/<имя_файла>.xml`. Имя файла может быть произвольным, например `integers.xml`. В коде приложения можно ссылаться на массивы целочисленных ресурсов для определения циклов, размеров файлов и пр.
- ✓ **Типизированный массив.** Используется для создания массивов ресурсов разных типов, например графических (`drawables`). Один массив может содержать ресурсы разных типов. Такие массивы называются *неоднородными*. При их использовании важно обеспечить правильное приведение типов, поскольку в одном и том же массиве типы могут быть разными. Типизированный массив может храниться в файле `/res/values/<имя_файла>.xml`. Имя файла может быть произвольным, например `types.xml`.

Меню

Строки меню можно определять либо в коде, либо в файле XML. Предпочтителен второй способ. Он более гибкий и мощный. Как это ни парадоксально, он в то же время менее трудоемкий. Меню, определенные в файлах XML, необходимо хранить в папке `menus/`. Каждое меню должно быть определено в отдельном файле XML.

Цвета

Файл цветов (например, `colors.xml`) должен быть расположен в папке `/res/values`. Определенные в нем цвета позволяют использовать в коде Java произвольные, заданные вами имена цветов, такие как `login_screen_font_color` (цвет_шрифта_в_регистрационном_окне). Каждый цвет определен как шестнадцатеричное значение.

Работа с ресурсами

В предыдущих главах вы уже несколько раз встречались с ресурсами. На данный момент вы уже знаете, что для доступа к ресурсам в приложении применяется автоматически генерируемый класс `R`. Если вы забыли об этом, прочитайте еще раз главу 3.

Перенос строк в ресурсы

Конечно, текстовые строки можно определять непосредственно в коде Java, для этого есть даже специальный тип `String`. Однако во многих случаях их лучше хранить в ресурсах. Фактически определять строки в коде целесообразно, только если нужно побыстрее создать простенькое приложение, во всех же других случаях рекомендуется хранить строки в ресурсах. Во время разработки приложения программист часто создает вспомогательные строки непосредственно в коде, рассчитывая на то, что после ряда экспериментов большинство из них будет удалено, а в ресурсы будут перенесены только строки, оказавшиеся полезными. Ниже рассмотрены два способа переноса текстовых строк из кода Java в ресурсы с помощью встроенных инструментов Eclipse.

Трудоемкий способ

Чтобы перенести строку из кода в ресурс, выполните следующие операции.

1. Создайте строковый ресурс и присвойте ему имя и значение.
2. Скопируйте имя строкового ресурса.
3. Замените строковое значение в компоновке или коде именем (идентификатором) строкового ресурса.

Вас может удивить, почему я назвал такую простую операцию трудоемкой. Ведь она занимает всего 30–45 секунд для разработчика средней квалификации. Однако существует еще более быстрый способ решения этой же задачи.

Быстрый способ

Этот способ переноса строки в ресурс занимает не более 15 секунд. Когда вы интенсивно работаете над приложением, вам придется выполнять эту операцию не менее 30 раз в день. Таким образом, вы сэкономите 15 минут только на операциях копирования и вставки. Ниже приведен быстрый способ переноса. Выполните следующие операции.

1. **В рабочей среде Eclipse откройте файл `main.xml`, расположенный в папке `layouts`.**
2. **Найдите в нем текстовую строку, которую нужно перенести в ресурс (ниже она отмечена полужирным шрифтом).**

```
<Button
   android:id="@+id/toggleButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center_horizontal"
   android:text="Переключить режим звонка"
/>
```

3. **Выделите строку, которую нужно перенести в ресурс.**

4. **Нажмите клавиши <Shift+Alt+A>.**
   Будет открыто меню с тремя пунктами.
5. **Выберите в меню команду Extract Android String (Извлечь строку Android).**
   Активизируется диалоговое окно Extract Android String (рис. 6.1). В нем можно устанавливать параметры создаваемого ресурса. Пока что не будем их рассматривать, поэтому просто щелкните на кнопке OK.
   Содержимое файлов `main.xml` и `strings.xml` автоматически изменится. В файле `main.xml` текстовая строка `Переключить режим звонка` будет заменена идентификатором ресурса `@string/switch_silent_mode`. Откройте файл `strings.xml`, расположенный в папке `res/values`, и вы увидите в нем текстовую строку `Переключить режим звонка` с заданным вами именем `switch_silent_mode`.

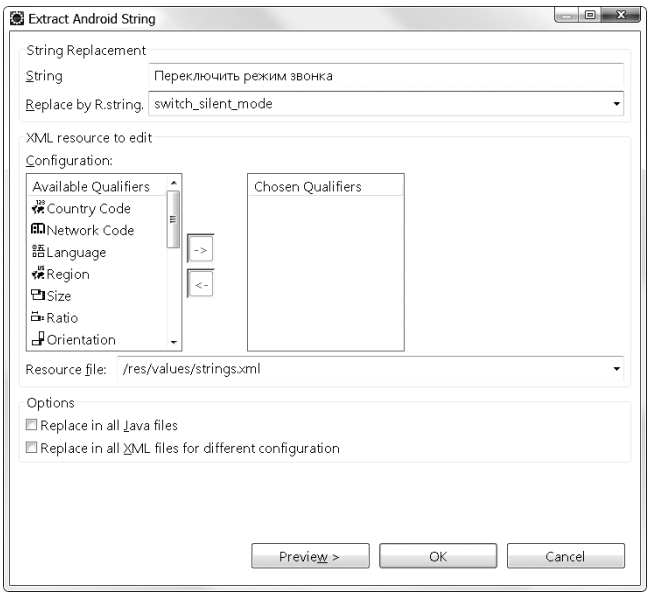


Рис. 6.1. Перенос текстовой строки в ресурс

Выше приведен пример переноса строки из файла XML. Перенос из файла Java выполняется точно так же. Вся операция занимает не более 15 секунд. Если вы выполняете ее многократно, то сэкономите довольно много времени. Но есть еще одно преимущество данного способа: выполняя перенос вручную, вы можете ошибиться, вводя идентификатор или текст, в то время как инструмент переноса выполняет эту операцию безошибочно.

Оптимизация изображений

Среди всех типов ресурсов тяжелее всего работать с изображениями, потому что качество изображения существенно зависит от разрешения экрана. Изображение может прекрасно выглядеть на экране со средним разрешением, но при низком разрешении это же изображение выглядит как мусор. Чтобы облегчить решение этой

проблемы, в Android определены отдельные папки ресурсов изображений для экранов с разными разрешениями (если вы забыли об этом, прочитайте еще раз главу 3).

Уменьшение и увеличение изображений

В приложение нужно заложить набор изображений, предназначенных для разных разрешений экрана. Сначала разрабатывайте графику для самой высокой разрешающей способности, например 300dpi, в файле с большими размерами. Для значка запуска приложения создайте рисунок 250×250px. Для папки `hdpi` нужен значок всего лишь 72×72px (сейчас такие значки используются практически всегда), но никто не гарантирует, что через полгода не появятся планшеты Android или Google TV с большими разрешающими способностями

Для работы с большими изображениями необходим современный мощный компьютер. Работать с ними на маломощном компьютере в принципе можно, но это значительно тяжелее. Тем не менее я рекомендую всегда начинать с высоких разрешающих способностей. Имея "сырой" файл изображения с высоким разрешением, всегда легче преобразовать его в файл с низким разрешением, чем наоборот.

Когда вы уменьшаете разрешение изображения в графическом редакторе, то видите, исказилось ли оно, и, если исказилось, можете принять меры. Если позволить делать это операционной системе, она автоматически уменьшит разрешение, не обращая внимания на качество рисунка. Если же вы начнете работу над изображением с низкого разрешения, то при его увеличении появится зернистость. Кроме того, при высоком разрешении вы не получите более качественного рисунка, чем при низком.

Использование слоев

Если вы создаете изображения в графическом редакторе, поддерживающем слои, я настоятельно рекомендую размещать отдельные элементы изображения на разных слоях. Такой подход предоставляет много преимуществ. Ниже рассмотрены два основных.

- ✓ **Изменения.** Часто возникает необходимость изменить что-либо в изображении, например фон, шрифт или логотип. Если эти элементы изображения расположены на разных слоях, любой из них легко будет изменить, не затрагивая другие. Если же все они находятся на одном слое, вам придется полностью переделывать изображение.
- ✓ **Локализация.** Ресурсы позволяют подставлять разные строки при разных языковых предпочтениях устройства. То же справедливо и для графики. В изображениях часто используются стилизованные текстовые элементы. Например, если ваше приложение, в изображениях которого встречается стилизованный русский текст, будет продаваться в Японии, создайте две папки ресурсов: `res/drawable-ru` и `res/drawable-ja`. Платформа Android распознает регион (Россия или Япония) и подставит в приложение нужный вариант изображения.

Локализация приложения с помощью ресурсов

Популярность операционной системы Android быстро увеличивается. Несмотря на то что она появилась всего несколько лет назад, по количеству продаж она уже обогнала многих своих конкурентов, включая iPhone. Поэтому пользователи Android есть во всех уголках Земного шара, и разговаривают они, естественно, на разных языках.

Что это означает для нас как разработчиков приложений Android? Это означает, что рынок приложений Android в данный момент предоставляет широчайшие возможности и практически неисчерпаем, причем наиболее перспективны локализованные приложения, в которых ресурсы используются для настройки приложения применительно к разным языкам и регионам. Предположим, ваше приложение написано на английском языке. Следовательно, с ним могут работать пользователи в Соединенных Штатах и Великобритании. Что произойдет, когда появится потребность распространять приложение, например, в Латинской Америке? Если вы жестко закодировали строковые значения в представлениях и деятельностях, то для подготовки релиза на испанском языке вам придется найти в коде все строки на английском и переписать их на испанском. А если то же самое нужно сделать еще и для португальского языка, чтобы продавать приложение в Бразилии? Вам опять придется найти все строки, причем и в первом, и втором случаях вы наверняка пропустите многие строки, и пользователи будут недовольны, встретив слова на незнакомом языке. Оптимальное решение данной проблемы состоит в использовании ресурсов. Вы создаете отдельную папку для каждого языка путем копирования исходной папки и нанимаете лингвиста, который переводит все строки в данной папке на соответствующий язык.

Ресурсы позволяют извлечь все читаемые человеком строки (включая тексты в изображениях) в отдельную папку, на которую можно ссылаться в приложении. Операционная система автоматически выберет нужную папку в соответствии с языком, используемым в данном регионе. Кроме того, разные папки ресурсов можно использовать для разных разрешений экрана и параметров компоновки (например, в зависимости от ориентации экрана — альбомная или портретная).

Если вы хотите, чтобы ваше приложение можно было использовать в разных странах, разместите все строки в ресурсах, а не в коде. Я рекомендую всегда размещать строки в файле `strigns.xml`, потому что когда-нибудь обязательно наступит момент, когда кто-то в другой стране захочет перевести ваше приложение на другой язык. Тогда вам нужно будет всего лишь нанять лингвиста, который переведет все строки в файле `strings.xml`, и создать папку ресурсов для этого языка. Например, если пользователи находятся в Китае и в их мобильных телефонах установлен китайский набор символов, операционная система Android автоматически найдет папку `values-cn`, в которой находится китайская версия файла `strings.xml`. Если Android не найдет нужную папку, будет применена папка `values`, в которой находится исходная версия `strings.xml`. Если вы по неопытности уже разместили строки в коде Java, выше было описано, как перенести их в файл ресурсов.

Вы должны выработать в себе привычку размещать все строки в ресурсах, а не в коде. Вы ведь планируете стать опытным разработчиком приложений Android, не так ли? Следовательно, вы неизбежно будете заняты разработкой локализованных приложений, продаваемых во многих странах.

Разработка локализованных приложений — большая и весьма сложная тема. Дополнительную информацию по этому вопросу можно найти в документации SDK:

```
http://developer.android.com/guide/topics/resources/
localization.html
```

Если необходимость быть готовым поставлять свое приложение в разные страны звучит для вас устрашающе, некоторым утешением послужит то, что сайт Android Market позволяет определять целевые регионы для вашего приложения. Вы не обязаны сделать приложение пригодным для поставки в любую точку Земного шара, тем не менее вам выгоднее, чтобы приложение можно было поставлять в как можно большее количество стран. Торговый сайт Android Market компании Google подробно рассматривается в главе 8.

Глава 7

Размещение виджетов на главном экране

В этой главе...

- Виджеты приложения в Android
- Отложенные намерения
- Создание виджета приложения на главном экране
- Добавление виджета на главный экран

Наиболее важное свойство приложения, влияющее на объем его продаж, — удобство работы с ним. Если работать с приложением неудобно, пользователи не будут его покупать.

Вы создали приложение Silent Mode Toggle, и оно работает прекрасно. Тем не менее если опубликовать его на сайте Android Market, оно будет не очень популярным по той причине, что с ним неудобно работать. Чтобы переключить телефон в режим бесшумного вызова, пользователь должен сначала открыть приложение Silent Mode Toggle, а затем нажать кнопку Переключить режим звонка. Если пользователь не создал ярлык приложения на главном экране и приложение "спрятано" на панели запуска наряду с полусотней других приложений, то для переключения режима звонка нужно выполнить ряд дополнительных операций: разблокировать телефон, открыть панель запуска приложений, найти на ней приложение Silent Mode Toggle, открыть это приложение и, наконец, нажать кнопку Переключить режим звонка. Для пользователя легче уменьшить громкость звонка с помощью кнопки громкости, которая есть почти на каждом мобильном телефоне. Как же сделать приложение Silent Mode Toggle более удобным? Очень просто! Нужно создать для него виджет на главном экране.

В этой главе я продемонстрирую создание виджета приложения на главном экране. Обычно виджеты приложений похожи на маленькие значки или очень маленькие представления, расположенные на главном экране. Такой виджет позволяет пользователю взаимодействовать с приложением путем прикосновения к значку (т.е. к виджету на главном экране). Когда пользователь прикасается к значку, включается базовая функция приложения, и оно переключает режим звонка. В данной главе будут рассмотрены следующие классы:

- ✓ `Intent`;
- ✓ `BroadcastReceiver`;
- ✓ `AppWidgetProvider`;
- ✓ `IntentService`;
- ✓ `AppWidgetProviderInfo`.

Каждый из этих классов играет важную роль не только в операционной системе Android, но и в инфраструктуре виджетов приложения.

Виджеты приложения в Android

В Android *виджеты приложения* — это миниатюрные приложения, которые можно внедрять в другие приложения, такие как главный экран (это тоже приложение Android). В подобном случае они называются *виджетами главного экрана*. Виджет приложения может принимать ввод символов пользователем посредством событий щелчка. Кроме того, он может обновлять свой внешний вид при необходимости. Виджеты приложения можно добавить на главный экран, выполнив длинное нажатие главного экрана (т.е. подержав палец более двух секунд). При этом активизируется диалоговое окно выбора виджетов (рис. 7.1).

Рис. 7.1. Диалоговое окно, появляющееся после длинного нажатия главного экрана

Чтобы сделать приложение Silent Mode Toggle более удобным для использования, создадим для него виджет, пригодный для размещения на главном экране. Нажав значок виджета, пользователь изменит режим звонка с громкого на бесшумный или наоборот, не открывая приложение Silent Mode Toggle. Кроме того, виджет должен автоматически обновлять свою компоновку, чтобы проинформировать пользователя о текущем режиме звонка (рис. 7.2).

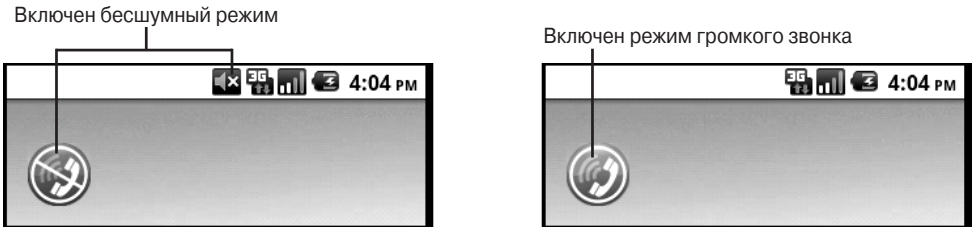


Рис. 7.2. В данной главе вы создадите виджет приложения с двумя этими состояниями

Дистанционные представления

При работе с операционной системой Android не следует забывать, что она основана на ядре Linux 2.6. В Linux применяется собственная стратегия обеспечения безопасности, а платформа Android наследует ее от Linux. Модель безопасности Android базируется на параметрах безопасности пользователей, файлов и процессов.

Каждое приложение Android ассоциировано (по крайней мере, должно быть ассоциировано) с уникальным идентификатором пользователя. Все процессы приложения выполняются от имени этого пользователя. Такая стратегия безопасности предотвращает изменение файлов одного приложения другим приложением, что могло бы привести к тому, что злоумышленник сможет внедрять свои коды в чужие приложения.

Главный экран — это фактически приложение, выполняющееся в данный момент на устройстве Android. Это приложение хостируется не вами, а операционной системой Android. Поэтому вам, как разработчику других приложений, не разрешено изменять фактически выполняющийся код главного экрана, потому что если бы разработчикам было позволено это делать, злоумышленники получили бы легкую возможность внедрять зловредные коды в приложения и операционную систему. Как же вам, добросовестному разработчику, можно изменить главный экран? Этому вопросу посвящены следующие разделы данной главы.

Разработчики операционной системы Android предоставляют вам, разработчику приложений Android, доступ к определенной области главного экрана посредством вашего же приложения. Проблема решается путем реализации архитектуры `RemoteViews` (Дистанционные представления), которая позволяет выполнять код внутри вашего приложения и за пределами приложения главного экрана. Этот код может изменять представление, расположенное на главном экране. Конечный результат состоит в том, что никакой дополнительный код не может выполняться в приложении главного экрана, а код виджета приложения выполняется только в вашем приложении.

Возможно, понятие виджета приложения сбивает вас с толку, поэтому представьте себе следующий сценарий. Пользователь прикасается к виджету приложения главного экрана (т.е. к значку на главном экране). Это действие инициирует запрос на изменение режима звонка. Запрос направляется в ваше приложение операционной системой Android, и оно (ваше приложение) обрабатывает запрос. Во время обработки запроса ваше приложение приказывает операционной системе Android изменить режим звонка и обновить значок виджета приложения на главном экране. Новое изображение

значка сообщит пользователю о том, что режим звонка изменился. Никакой фрагмент этого кода не выполняется в приложении главного экрана, весь он выполняется удаленно в вашем приложении, а операционная система Android маршрутизирует сообщения в соответствующие приложения.

Дистанционные представления с одной стороны — нечто магическое, а с другой — инновационная технология. Базовый класс такого представления `RemoteViews` позволяет приложению программно предоставить удаленный пользовательский интерфейс главному экрану в другом процессе. Код виджета приложения является не деятельностью (как в предыдущих главах), а реализацией класса `AppWidgetProvider` (провайдер виджета приложения). Операционная система Android маршрутизирует сообщения соответствующему приложению. Когда она маршрутизирует сообщения в ваше приложение с главного экрана, сообщения обрабатываются реализацией класса `AppWidgetProvider`.

Использование класса `AppWidgetProvider`

Класс `AppWidgetProvider` предоставляет точки подключения, позволяющие программно взаимодействовать с виджетом приложения на главном экране. Когда пользователь взаимодействует с виджетом приложения, между виджетом приложения главного экрана и приложением передаются сообщения посредством широковещательных событий. Эти события позволяют реагировать на обновление, включение, отключение и удаление виджета приложения. Можно также обновлять внешний вид и поведение виджета приложения на главном экране путем подключения нового представления. Это представление размещено на главном экране, а не в фактически выполняющемся приложении, поэтому для обновления компоновки главного экрана используется класс `RemoteViews`. Вся логика взаимодействия инициализируется путем реализации класса `AppWidgetProvider`.

Инфраструктуру виджета приложения можно представлять себе как “переводчика” в канале взаимодействия двух сущностей. Предположим, вам нужно поговорить с человеком, который знает итальянский язык, но сами вы не знаете итальянский. Как быть? Естественно, нужно найти переводчика. Он слушает ваши слова, переводит их на итальянский и говорит перевод человеку, с которым вы общаетесь. Так же работает инфраструктура виджета приложения: она ваш переводчик.

Данная аналогия с переводчиком более глубокая, чем может показаться на первый взгляд. Когда человеку, говорящему по-итальянски (главному экрану), нужно сообщить вам, что кое-что произошло (например, что нажата кнопка), переводчик (инфраструктура виджета приложения и операционная система Android) преобразует его слова в системное сообщение, которое вы можете понять. Получив сообщение, вы можете отреагировать на него, как запрограммировано (например, принять решение изменить цвет фона виджета приложения на зеленый), и переводчик преобразует и передаст ваше сообщение человеку, говорящему по-итальянски (главному экрану). После этого главный экран обновит представление виджета, сделав его фон зеленым.

Обновление представления — это почти единственное, что можно сделать с виджетом приложения. Не менее мощное ограничение наложено и с обратной стороны: виджет приложения может принимать только прикосно-

вение и генерировать событие прикосновения (щелчка). Работая с виджетом приложения, программист не имеет доступа к другим базовым виджетам ввода, таким как редактируемые текстовые поля, раскрывающиеся списки и любые другие средства ввода информации.

Отложенные намерения

Когда пользователю нужно начать работать с приложением, он делает это посредством операционной системы с использованием архитектуры сообщений Android. Поэтому после прикосновения к виджету приложение не получает сообщение об этом немедленно. Впрочем, это не означает, что приложение не может быть оповещено о щелчке.

Событие щелчка на виджете приложения содержит инструкцию о том, что нужно сделать. Эта инструкция передается посредством класса `PendingIntent` (Отложенное намерение) инфраструктуры Android. Класс `PendingIntent` является реализацией класса `Intent`.

Система намерений Android

Намерения — важный компонент инфраструктуры Android. Чтобы лучше понять, что такое намерение, представьте себе, что вы зажигаете свет в комнате с помощью выключателя. Ваше намерение — зажечь свет, ваше действие — щелчок выключателя. В Android это означает создание объекта `Intent` (Намерение) с параметром `Action` (Действие), определяющим, что должно быть сделано.

```
Intent turnLightOn = new Intent("Включить_свет");
```

Намерение передается в систему сообщений Android, а соответствующее действие (или несколько действий, т.е. объектов класса `Action`) обрабатывает объект `Intent` (если задано несколько действий, Android предоставит пользователю возможность выбрать нужное). В физическом мире электрическое соединение создается путем позиционирования контакта в корпусе выключателя, а в Android программист должен создать код действия, делающий то, что нужно сделать. На языке Android говорят, что "намерение обрабатывается действием". Это означает, что действие реагирует на получение намерения. При работе с виджетом приложения намерение нужно обработать в классе `BroadcastReceiver` (Широковещательный приемник). Объект `AppWidgetProvider` представляет собой экземпляр класса `BroadcastReceiver` с несколькими дополнительными средствами, реализующими инфраструктуру виджета приложения. Объект `BroadcastReceiver` отвечает за получение широковещательных сообщений.

Объект `AppWidgetProvider` — это "переводчик". Следовательно, `AppWidgetProvider` обрабатывает намерение, полученное от главного экрана, и реагирует на него соответственно тому, как вы закодировали это в пользовательском объекте `AppWidgetProvider`. Однако `AppWidgetProvider` работает не с любым намерением. Чтобы получать информацию от виджета приложения, необходимо применить отложенное намерение `PendingIntent`.

Объект `PendingIntent` содержит дочерний объект `Intent`. На высоком уровне отложенное намерение работает так же, как обычное. Чтобы понять, что такое

отложенное намерение, необходимо глубже ознакомиться с его базовым классом `Intent`. Намерение — это сообщение, которое может переносить "в себе" данные, описывающие требуемую операцию. Намерение может быть адресовано определенной деятельности или обобщенной категории получателей типа `BroadcastReceiver`, т.е. объектов `AppWidgetProvider`. Система классов `Intent`, `Activity` и `BroadcastReceiver` похожа на архитектуру магистрали сообщений, в которой сообщение передается на магистраль, после чего одна или несколько конечных точек магистрали реагируют на сообщение, только если они знают, что с ним делать. Если все конечные точки магистрали отказываются от сообщения или сообщение не адресовано конечной точке, система игнорирует его.

Намерение может быть запущено в магистраль сообщений несколькими способами.

Если для запуска новой деятельности используется вызов метода `startActivity()`, этот метод может получать объект `Intent` в качестве параметра.

Для извещения компонентов `BroadcastReceiver` можно использовать вызовы метода `sendBroadcast()`, принимающего намерение в качестве параметра.

Для взаимодействия с фоновой службой (см. далее) можно использовать методы `startService()` и `bindService()`, которые тоже принимают намерения в качестве параметров.

Деятельности можно рассматривать в качестве "клея", связывающего разные компоненты приложения, потому что они реализуют механизм позднего связывания, обеспечивающий взаимодействие компонентов внутри и за пределами приложения.

Данные намерений

Намерение содержит следующую информацию.

- ✓ **Действие.** Операция, которая должна быть выполнена при получении намерения. Наиболее популярные действия — `ACTION_VIEW`, `ACTION_EDIT` и `ACTION_MAIN`. Кроме того, при необходимости программист может создать произвольные действия.
- ✓ **Данные.** Информация, обрабатываемая при получении намерения, например запись в базе данных, URL-адрес страницы, которую нужно открыть, и т.п.

В табл. 7.1 в качестве примера приведены несколько параметров действий и простых структур данных объектов `Intent`.

Таблица 7.1. Примеры данных намерения

Действие	Данные	Результат
`ACTION_VIEW`	`tel:123`	Отображение номеронабирателя с подставленными числами `123`
`ACTION_DIAL`	`content://contacts/people/1`	Отображение номеронабирателя с подставленным телефонным номером контакта, имеющего идентификационный номер 1
`ACTION_EDIT`	`content://contacts/people/1`	Редактирование информации о контакте 1

Окончание табл. 7.1

Действие	Данные	Результат
ACTION_VIEW	http://www.example.org	Отображение веб-страницы данного намерения
ACTION_VIEW	content://contacts/people	Отображение списка лиц в системе хранения контактов

Намерения могут переносить и другие данные, включая следующие.

- ✓ **category:.** Информация о выполняемом действии. Например, код CATEGORY_LAUNCHER означает, что приложение должно присутствовать в меню запуска приложений на верхнем уровне. При значении CATEGORY_ALTERNATIVE могут выполняться альтернативные действия над пользовательскими данными.
- ✓ **type:.** Задание определенного типа MIME данных намерения. Например, при задании типа audio/mpeg операционная система Android распознает данные как файл MP3. Обычно тип данных легко извлечь из самих данных. Если же задать тип MIME, то он переопределит тип, извлеченный из данных. Следовательно, явно заданный тип переопределяет неявно заданный.
- ✓ **component:.** Задание явного имени компонента для класса, в котором выполняется намерение. Обычно компонент можно извлечь путем просмотра другой информации, заложенной в намерении (действие, тип данных, категория и др.). Извлеченный компонент обрабатывает информацию намерения. Если атрибут компонента установлен, извлечение не выполняется, и компонент используется точно как задано в намерении. Такой сценарий используется в приложениях чаще всего, однако программист может предоставить в качестве компонента другую деятельность, чтобы приказать Android взаимодействовать с определенным классом.
- ✓ **extras:.** Набор дополнительной информации на основе ключей. Используется для передачи данных в принимающий компонент. Например, если нужно передать электронный адрес, можно использовать данные extras: для передачи тела, темы и других компонентов электронного письма.

Обработка намерений

Операционная система Android обрабатывает намерения одним из следующих двух способов.

- ✓ **Явно.** Намерение определяет явный компонент или точный класс, который будет выполнять нужные операции над данными намерения. Это наиболее распространенный способ обработки намерений. Намерения такого типа часто не содержат данных, поскольку они предназначены для запуска других деятельностей приложения. Пример использования явных намерений в приложении рассматривается далее.

- ✓ **Неявно.** Намерение не определяет компонент или класс, а предоставляет вместо этого достаточно информации о действии, которое должно быть выполнено операционной системой, или об определении доступного компонента или класса, который должен обработать намерение.

  Характерный пример неявного намерения — электронное письмо, содержащее поля адреса, темы, тела, вложения и типа MIME. Операционная система интерпретирует это намерение как электронное письмо и предоставляет пользователю возможность выбрать почтовое приложение, которое должно обработать это намерение. Возможные варианты — Gmail, Exchange или учетная запись POP, зарегистрированная в устройстве. Меню позволяет выбрать, откуда будет отправлено письмо. Процедура определения компонента или класса для данного намерения иногда называется *разрешением намерения*.

Использование отложенных намерений

Объект `PendingIntent` (Отложенное намерение) — это, в сущности, обычное намерение, но с немного другой функциональностью. Оно создается одним приложением и передается совершенно другому приложению. Передавая другому приложению отложенное намерение, вы передаете ему право выполнять операцию, которую вы создали в рамках и с правами первого приложения. Говоря языком дилетанта, вы передаете информацию о том, как вызвать ваше приложение для выполнения некоторой операции под именем другого приложения. Когда другое приложение решит, что операцию нужно выполнить, оно прикажет системе сообщений Android проинформировать ваше приложение о необходимости выполнить эту операцию.

Загрузка процессора

Объекты `RemoteViews` сильно загружают процессор и память. Кроме того, они приводят к быстрому разряду батареи. Это связано с тем, что система Android вынуждена переносить объекты `RemoteViews` через границы процессов. Поэтому при использовании объекта `RemoteViews` важно закладывать в него как можно меньше операций и завершать его как можно быстрее. Если приложение реагирует слишком долго, Android может сгенерировать для него ошибку ANR (Application Not Responding — приложение не отвечает). В отличие от настольных операционных систем, операционная система Android очень экономная. Если какому-либо приложению не хватает ресурсов, она может закрыть его, сгенерировав ошибку ANR. Предположим, приложение загружает обновление какой-либо сетевой службы, например Twitter. Если загрузка выполняется слишком долго, Android генерирует ошибку ANR и отображает на экране виджет, предлагающий пользователю решить, что нужно сделать: закрыть приложение принудительно или продолжить работу.

Один из способов избежать появления окон ANR состоит в реализации службы внутри объекта `AppWidgetProvider`. В следующих разделах вы ознакомитесь с реализацией службы `IntentService`, позволяющей избежать ошибки ANR при сохранении быстродействия виджета.

Для получения экземпляра отложенного намерения используется вызов `PendingIntent.getBroadcast()`. Этот метод возвращает объект `PendingIntent`, используемый для широковещательного распространения намерений по системе. При вызове данный метод принимает четыре параметра.

- ✓ `Context` — контекст, в котором объект `PendingIntent` должен выполнить широковещательную передачу.
- ✓ `RequestCode` — закрытый код запроса для получателя. В данном примере этот параметр не используется, поэтому передается нуль.
- ✓ `Intent` — передаваемое намерение.
- ✓ `Flags` — набор элементов управления, которые будут контролировать намерение после его запуска. В данном примере этот параметр не используется, и вместо него передается нуль.

Что-то много у нас намерений получилось. Чтобы создать объект `PendingIntent`, нужно иметь объект `Intent`? Именно так! Объект `Intent` заключается в оболочку `PendingIntent` для передачи через границу процессов (будто спрятали в чемодан для перевоза через границу). После создания объекта `PendingIntent` реальная работа выполняется дочерним объектом `Intent`.

Однако все это была теория. Теперь же перейдем к ее практическому применению в проекте Silent Mode Toggle для создания виджета приложения.

Создание виджета приложения на главном экране

При взаимодействии пользователя с виджетом приложения на главном экране в игру вступают многие компоненты Android. Система сообщений Android управляет процессом обмена сообщениями между виджетом приложения на главном экране и вашим приложением. Свою роль играют также объекты `PendingIntent` и `AppWidgetProvider`. В данном разделе я продемонстрирую, как создать каждый компонент таким образом, чтобы вы смогли разместить свой первый виджет приложения на главном экране и запустить его путем прикосновения к экрану.

Реализация объекта `AppWidgetProvider`

Чтобы создать объект `AppWidgetProvider`, откройте рабочую среду Eclipse, а в ней — приложение Silent Mode Toggle.

Добавьте новый класс в пакет `com.dummies.android.silentmodetoggle` и присвойте ему произвольное имя. Чтобы оно везде совпадало с упоминаемым далее, присвойте ему имя `AppWidget.java`. Для добавления нового класса щелкните правой кнопкой мыши на папке `src/com.dummies.android.silentmodetoggle` приложения Silent Mode Toggle и выберите в контекстном меню команду New⇨Class (Создать⇨Класс). Активизируется диалоговое окно New Java Class (Новый класс Java). Введите имя класса `AppWidget` и задайте базовый класс `android.appwidget.AppWidgetProvider` (рис. 7.3). Щелкните на кнопке Finish (Готово). Новый класс будет добавлен в указанный пакет, и на правой панели Eclipse откроется содержимое файла `AppWidget.java`.

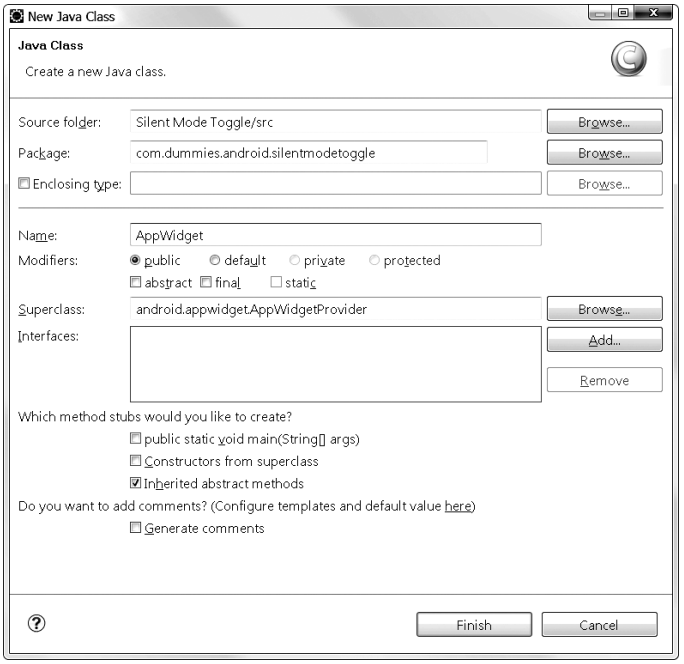


Рис. 7.3. Создание класса `AppWidget`

Класс `AppWidgetProvider` делает всю работу, необходимую для реагирования на события дистанционного представления `RemoteViews`. Но как он это делает? Если посмотреть в документацию класса `AppWidgetProvider`, то можно увидеть, сто он является прямым потомком класса `BroadcastReceiver` (приемник широковещательных сообщений). На высоком уровне объект `BroadcastReceiver` представляет собой компонент, который может принимать широковещательные сообщения операционной системы Android. Когда пользователь прикасается к экземпляру представления `RemoteViews` на главном экране (например, к значку), Android генерирует широковещательное сообщение, информирующее приемник сообщения о том, что произошел щелчок. В Android сообщение направляется определенному целевому объекту. Когда сообщение передано, объект `AppWidgetProvider` может обработать его.

Обратите внимание на то, что эти сообщения широковещательные, т.е. они рассылаются по всей системе. Если содержимое и адресная информация сообщения достаточно расплывчатые, сообщение может быть обработано многими объектами `BroadcastReceiver`. Однако в данном примере создаваемый нами объект `AppWidgetProvider` предназначен для единственного целевого объекта. Представьте, будто вы входите в комнату, в которой сидят строительные подрядчики, и спрашиваете, кто может построить для вас дом. Все они отреагируют, ответив положительно. Если вы спросите о доме определенного типа, отреагирует только часть подрядчиков. Если же вы назовете имя нужного вам подрядчика, отреагирует только он (естественно, если он есть в комнате). Последний случай — пример подробной детализации адреса и содержимого сообщения.

Взаимодействие с виджетом приложения

Сейчас в вашем классе `AppWidget` нет кода, это всего лишь заготовка. Чтобы класс делал полезную работу, нужно добавить в него код, реагирующий на намерение, переданное базовому классу `AppWidgetProvider`. В файле `AppWidget.java` введите код, приведенный в листинге 7.1. Номера строк справа, как обычно, не вводите; они добавлены для комментирования кода.

Листинг 7.1. Инициализация виджета приложения

```
public class AppWidget extends AppWidgetProvider {                    1

  @Override
  public void onReceive(Context ctxt, Intent intent) {                 4
    if (intent.getAction()==null) {                                    5
    // Выполнить какие-нибудь действия
    } else {                                                           8
      super.onReceive(ctxt, intent);                                  10
    }
    }

  @Override
  public void onUpdate(Context context, AppWidgetManager
    appWidgetManager, int[] appWidgetIds) {                          15
    // Выполнить какие-нибудь действия
  }
}
```

Ниже приведено краткое объяснение работы отмеченных строк.

- ✓ **1.** Эта строка кода информирует систему о том, что ваш класс `AppWidget` наследует базовый класс `AppWidgetProvider`.
- ✓ **4.** Переопределение метода `onReceive()`, позволяющее перехватывать намерения, полученные от `RemoteViews`. Данное намерение может быть инициировано пользователем, прикоснувшимся к представлению, что равносильно щелчку на кнопке. Объект `Intent` находится в объекте `PendingIntent`, который инициировал запрос.
- ✓ **5.** Как уже упоминалось, объект `Intent` может содержать разные части данных. Одна из таких частей данных — действие. Эта строка кода проверяет, есть ли действие в намерении. Если действия нет, значит, это наше намерение и нужно отреагировать на него.
- ✓ **8.** Действие есть. Следовательно, произойдет ряд событий и будут выполнены некоторые операции.
- ✓ **10.** Работа делегируется в базовый класс. Делать что-либо с намерением не нужно, потому что мы ожидали намерение без действия. Приложение перейдет на данную ветвь кода, когда виджет приложения регулярно обновляет себя автоматически, что будет определено в метаданных виджета (см. далее). Переход в базовый класс приведет к вызову одного из многих встроенных методов, которые включают, отключают, запускают, останавливают или обновляют (см. строку 15) виджет приложения.

- ✓ **15.** Метод `onUpdate()` вызывается инфраструктурой Android по расписанию, установленному в метаданных виджета. Обычно этот метод вызывается, когда нужно обновить представление без участия пользователя. Классический пример — виджет приложения новостей, который нужно обновлять каждые 30 минут, подставляя новейшие заголовки статей. Пользователю ничего делать не нужно, так как виджет регулярно обновляет сам себя.

Компоновка виджета приложения

Чтобы операционная система Android могла выяснить, как виджет должен быть отображен на главном экране, виджет приложения должен быть определенным образом скомпонован. Файл компоновки виджета определяет, как он будет выглядеть на главном экране. Ранее (см. рис. 7.2) приводились два снимка главного экрана с виджетом, выполняющимся в эмуляторе. Значки на снимках определены в файле компоновки виджета. Фон значков прозрачный, поэтому на рис. 7.2 прямоугольная рамка значка не видна. Если в файле компоновки виджета изменить цвет фона с прозрачного на любой другой, не совпадающий с фоном главного экрана (например, на светло-зеленый), вокруг значка будет видна прямоугольная рамка (рис. 7.4).

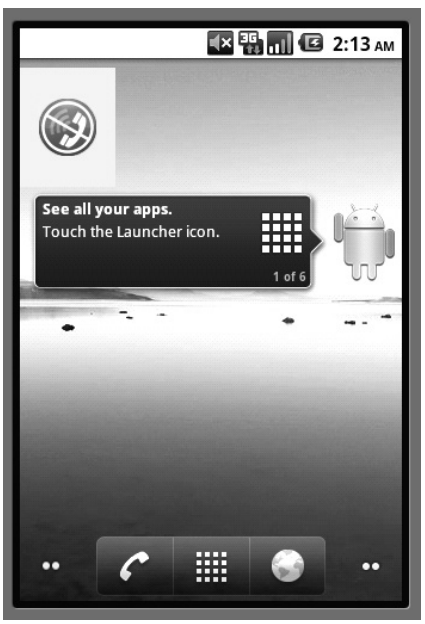


Рис. 7.4. Результат изменения цвета фона виджета

Сделав фон виджета непрозрачным, можно проиллюстрировать концепцию экранного пространства виджета. Светло-зеленый прямоугольник на рис. 7.4 идентифицирует экранное пространство, доступное для виджета приложения. Виджет может занять одну или несколько ячеек на главном экране. В данном случае (см. рис. 7.4) виджет занял одну ячейку.

Чтобы определить компоновку виджета, создайте файл XML в папке `res/layouts`. Присвойте файлу компоновки имя `widget.xml`. Введите в файл `widget.xml` содержимое, приведенное в листинге 7.2.

Листинг 7.2. Содержимое файла компоновки `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/icon"
    android:clickable="true" />                                    9
</RelativeLayout>
```

В этой компоновке нет ничего такого, что вы еще не видели. В контейнер `RelativeLayout` вложено одно дочернее представление `ImageView`, чувствительное к щелчку. Свойство чувствительности к щелчку задано значением `true` атрибута `clickable` в строке 9 листинга 7.2.

Обратите внимание на свойство `src` элемента `ImageView`, которому присвоен значок приложения. Возможно, вам это покажется странным (ведь отображаться должен не он, а изображение с телефонной трубкой), поэтому нужно объяснить, зачем я так сделал. Когда я создавал компоновку, кнопок, представляющих громкий и бесшумный режимы, еще не существовало. Однако при создании компоновки нужно видеть, как она будет выглядеть на экране. Поэтому я подставил значок, который "был под рукой". Тот факт, что подставлен значок, который не будет отображаться, ни на что не влияет. Когда виджет приложения будет загружен, класс `ToggleService` переключит значение `src` на значок бесшумного или громкого режимов (см. далее).

Значок приложения на главном экране позволяет с первого взгляда увидеть текущее состояние приложения. Значок `phone_state_normal` виден, когда звонок телефона переключен в громкий режим, а значок `phone_state_silent` — когда включен бесшумный режим. Я создал эти значки в графической программе, позволяющей редактировать изображения.

Выполнение нужных операций в объекте `AppWidgetProvider`

После того как намерение запустило объект `AppWidgetProvider`, он должен выполнить некоторую работу от имени вызывающего приложения (в данном случае — приложения главного экрана). В следующих разделах я покажу, как это делается.

Прежде чем приступить к кодированию, необходимо понять, как работа будет сделана в `AppWidgetProvider`. Все операции лучше выполнить в фоновой службе, потому что, во-первых, они интенсивно нагружают систему, а во-вторых, они выполняются в другом процессе. Поэтому в данном примере изменение режима звонка выполняется в фоновой службе.

Класс `IntentService`

Зачем применять фоновую службу для такой простой операции, как изменение режима звонка? В данном разделе я отвечу на этот вопрос.

Любой код, который выполняется слишком долго, не отвечая на запросы Android, может быть прерван операционной системой путем генерирования ошибки ANR (Application Not Responding — приложение не отвечает). Виджеты приложений особенно "притягательны" к ошибкам ANR, потому что они выполняют код в удаленном процессе. Это значит, что они должны не только выполнить код, но и пройти через границы процессов, потратив время на запуск и инициализацию нового процесса, выполнение кода и уничтожение процесса. Все это потребляет много ресурсов процессора и памяти. Операционная система Android "присматривает" за виджетами приложений и не позволяет им выполняться слишком долго, в противном случае главный экран и другие приложения будут заблокированы на определенное время, ощутимое для пользователя, в результате чего он будет недоволен работой устройства. Поэтому Android следит за тем, чтобы вы, как программист, не могли "заморозить" устройство более чем на несколько секунд.

Поскольку виджеты приложения сильно загружают процессор и память, всегда тяжело судить, будут ли они порождать ошибки ANR. Если на устройстве не выполняются другие задачи, интенсивно потребляющие ресурсы, то виджет приложения, скорее всего, будет работать удовлетворительно. Однако если нажать на виджет в момент, когда устройство занято трудоемкими операциями, реакция виджета приложения может замедлиться, в результате чего Android сгенерирует ошибку ANR. Неизвестное состояние процессора — фактор, опасный для виджета приложения. Следовательно, чтобы обойти этот фактор, рекомендуется перенести работу виджета приложения в службу `IntentService`, которая может выполнять необходимые операции так долго, сколько потребуется, не затрагивая приложения главного экрана.

В отличие от большинства фоновых служб, которые выполняются долго, в `IntentService` применяется очередь задач, в которой каждое намерение обрабатывается в рабочем потоке. Процедура обработки завершается, как только очередь опустеет. Ни одна задача не проходит "без очереди", и ни одна не стоит в очереди дольше других.

Реализация объектов `AppWidgetProvider` и `IntentService`

В классе `AppWidget` введите код, приведенный в листинге 7.3.

Листинг 7.3. Код класса `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("AppWidget$ToggleService");                           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);        52
    updateViews.setOnClickPendingIntent(
                           R.id.phoneState,pi);             54
    return updateViews;                                     56
    }
  }
}
```

Ниже приведено описание операций, выполняемых в отмеченных строках кода.

- ✓ **6.** Создание экземпляра службы `ToggleService`. Объект `context` в этой строке кода ссылается на объект `Context` операционной системы Android, являющийся интерфейсом глобальной информации о приложении. Контекст передается в методы `onReceive()` и `onUpdate()`. Новое намерение создается, чтобы Android знала, что должно произойти.

Метод `onReceive()` инициируется прикосновением пользователя к виджету приложения на главном экране.

- ✓ **16.** Та же операция, что и в строке 6, но для обновления виджета.
- ✓ **19.** Реализация фоновой службы `IntentService`, которая делает то же, что и `MainActivity` (т.е. переключает режим звонка), но с учетом инфраструктуры виджета приложения. Данный статический класс вложен в класс виджета приложения.
- ✓ **22.** Вызов конструктора базового класса, полезного для отладки потоков. Если опустить этот вызов, будет сгенерирована ошибка компиляции, сообщающая о том, что вы должны явно вызвать конструктор базового класса. Если виджет приложения назван как-либо иначе, нужно привести его имя в параметре метода.
- ✓ **26.** Метод `onHandleIntent()` обрабатывает намерение, полученное службой. В данном примере это намерение создано в строках 6–16. Поскольку намерение явное (т.е. задано имя выполняемого класса), дополнительные данные не предоставляются. Следовательно, после достижения строки 26 намерение больше не нужно. Однако можно предоставить дополнительную информацию объекту `Intent`, который был извлечен из параметра этого метода. В данном случае объект `Intent` — всего лишь носитель, передающий службе `ToggleService` приказ начать обработку намерения.
- ✓ **27.** Создание объекта `ComponentName`, используемого совместно с `AppWidgetManager` (см. далее) в качестве провайдера нового содержимого, которое будет передано виджету приложения посредством экземпляра `RemoteViews`.
- ✓ **28.** Экземпляр `AppWidgetManager` получен путем статического вызова `AppWidgetmanager.getInstance()`. Класс `AppWidgetManager` отвечает за обновление состояния виджета приложения и предоставляет необходимую информацию об установленном виджете приложения. Мы будем использовать его для обновления состояния виджета приложения.
- ✓ **29.** Виджет приложения обновляется путем вызова метода `updateAppWidget()`. Для вызова нужны объекты `ComponentName` и `RemoteViews`. Принадлежащий операционной системе Android объект `ComponentName`, созданный в строке 27, выполняет обновление виджета приложения. Объект `RemoteViews` используется для обновления состояния виджета приложения на главном экране.
- ✓ **30.** Определение метода `buildUpdate()`. Этот метод возвращает новый объект `RemoteViews`, который будет использован в строке 29. В данном методе определены действия и логика операций, выполняемых над виджетом приложения.
- ✓ **32.** Создание объекта `RemoteViews` с именем текущего пакета и компоновкой, возвращаемой методом `buildUpdate()`. Компоновка `R.layout.widget` приведена в листинге 7.3.

- ✓ **34.** Получение экземпляра `AudioManager` и проверка состояния звонка. Если сейчас режим звонка бесшумный, значит, пользователь, прикасаясь к значку приложения, хочет переключить его в громкий режим.
- ✓ **40.** Получив объект `RemoteViews`, необходимо обновить его. Здесь это делается так же, как в `MainActivity` в предыдущей главе. Объект `RemoteViews` изменяет представление `R.id.phoneState ImageView` на `R.drawable.phone_state_normal` (правый значок на рис. 7.2).
- ✓ **45.** Блок `else`, расположенный над этой строкой, обновляет изображение в `ImageView`, заменив его на `R.drawable.phone_state_silent`, потому что перед этим был включен громкий режим (пользователь хочет переключить телефон в бесшумный режим).
- ✓ **49.** Создание объекта `Intent`, который запустит класс `AppWidget`.
- ✓ **52.** Виджеты приложения не могут сообщаться с обычными намерениями. Для них необходимы отложенные намерения. Как вы помните, виджеты приложений реализуют взаимодействие процессов, следовательно, необходимы объекты `PendingIntent`. В данной строке создается объект `PendingIntent`, который приказывает виджету приложения выполнить следующее действие посредством дочернего намерения, созданного в строке 49.
- ✓ **54.** Поскольку код работает с объектом `RemoteViews`, необходимо повторно скомпилировать всю иерархию событий представления. Инфраструктура виджета приложения заменит весь объект `RemoteViews` новым представлением, полученным посредством данного метода. Следовательно, осталось только сообщить представлению `RemoteViews`, что оно должно сделать при нажатии значка на главном экране. Объект `PendingIntent`, созданный в строке 52, инструктирует виджет приложения, что нужно сделать при нажатии значка. Данная операция устанавливается методом `setOnClickPendingIntent()`. Этот метод принимает два параметра: идентификационный код представления (в данном случае изображения, к которому прикоснулся пользователь) и объект `pi`, содержащий отложенное намерение, созданное в строке 52. Иными словами, вы устанавливаете в виджете приложения приемник щелчка на `ImageView`.
- ✓ **56.** Возврат созданного `RemoteViews`, чтобы вызов `updateAppWidget()` в строке 29 мог обновить виджет приложения.

Скопируйте в папку `drawable-mdpi` два файла маленьких значков `phone_state_normal.png` и `phone_state_silent.png`. На левой панели Eclipse щелкните правой кнопкой мыши на папке `drawable-mdpi` и выберите в контекстном меню команду Refresh (Обновить). Будут автоматически сгенерированы два новых ресурса — `phone_state_normal` и `phone_state_silent`. По умолчанию ресурс получает имя файла без расширения `.png`.

Метаданные виджета приложения

Итак, код обновления виджета приложения у нас есть. Теперь нужно сделать так, чтобы виджет приложения Silent Mode Toggle появился в меню, отображаемом после длинного нажатия главного экрана. Для этого нужно добавить в проект еще один файл XML. В нем должны быть приведены базовые метаданные виджета приложения, чтобы система Android могла узнать, как следует разместить виджет приложения на главном экране. Выполните следующие операции.

1. **На левой панели Eclipse щелкните правой кнопкой мыши на папке `res` и выберите в контекстном меню команду New⇨New Folder (Создать⇨Новая папка).**
2. **В качестве имени новой папки введите `xml` и щелкните на кнопке Finish (Готово).**
3. **Щелкните правой кнопкой мыши на папке `res/xml` и выберите команду New⇨Android XML File (Создать⇨Файл Android XML).**
4. **Введите имя файла `widget_provider.xml`.**
5. **Установите переключатель AppWidget Provider (Провайдер виджета приложения), задающий тип файла. Щелкните на кнопке Finish.**
6. **Когда файл будет открыт в редакторе XML, щелкните на вкладке widget_provider.xml, чтобы открыть окно текстового редактирования. Введите следующий код в файл `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"
/>
```

Атрибуты `minWidth` и `minHeight` определяют минимальное пространство, выделяемое представлению на главном экране. Можете увеличить их значения, чтобы значки были крупнее.

Атрибут `updatePeriodMillis` определяет, как часто виджет приложения будет автоматически пытаться обновить себя. В приложении Silent Mode Toggle эта операция в принципе не нужна. Поэтому атрибуту присвоено значение в миллисекундах, эквивалентное 30 минутам. Следовательно, каждые 30 минут приложение будет пытаться обновить себя, передав намерение, которое в объекте `AppWidgetProvider` вызывает метод `onUpdate()`.

Атрибут `initialLayout` идентифицирует внешний вид виджета приложения, когда он впервые появился на главном экране и еще ничего не сделал. Обратите внимание на то, что для инициализации виджета приложения и обновления его объекта `RemoteViews` путем вызова метода `onReceive()` может понадобиться несколько секунд.

В реальности могут происходить еще более длительные задержки, например если виджет приложения запрашивает в Твиттере статус обновления. Если сетевое соединение медленное, компоновка `initialLayout` будет отображаться, пока не будет получено обновление от Твиттера. В таком случае рекомендуется закодировать в

объекте initialLayout сообщение для пользователя о том, что выполняется загрузка информации. Если пользователь будет ждать слишком долго, не видя, что происходит, он может подумать, что устройство испортилось. Чтобы избавить пользователя от неприятных ощущений, закодируйте в объекте initialLayout отображение представления TextView с надписью Выполняется загрузка, пока работает объект AppWidgetProvider.

Сейчас можете установить приложение Silent Mode Toggle, выполнить длительное нажатие на главном экране и выбрать категорию Widgets. В списке вы увидите приложение Silent Mode Toggle. Это произошло благодаря тому, что вы задали в метаданных включение приложения в список виджетов. Однако при попытке добавить виджет приложения на главный экран будет сгенерировано исключение, потому что файл ApplicationManifest.xml еще ничего не знает об объектах IntentService и BroadcastReceiver. Приложение не знает, где их найти.

Регистрация новых компонентов в манифесте приложения

Каждый раз, когда вы добавляете в приложение объект Activity, Service или BroadcastReceiver (как и любой другой компонент), его нужно зарегистрировать в файле манифеста приложения. Операционная система Android извлекает из манифеста приложения важную информацию о компонентах приложения. Объекты Activity, Service и BroadcastReceiver, не зарегистрированные в манифесте приложения, не распознаются системой и, следовательно, не могут быть запущены. Если добавить виджет приложения на главный экран, он потерпит крах, потому что объект AppWidgetProvider является приемником BroadcastReceiver, причем в коде приемника используется служба, тоже не зарегистрированная в манифесте.

Чтобы добавить объекты AppWidgetProvider и IntentService в файл манифеста приложения, откройте в редакторе Eclipse файл AndroidManifest.xml и введите в существующий файл код, приведенный в листинге 7.4. Полужирным шрифтом отмечены новые строки, регистрирующие указанные компоненты.

Листинг 7.4. Файл AndroidManifest.xml с новыми компонентами

```
<?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/icon"
    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/icon">                          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" />
  </application>
  <uses-sdk android:minSdkVersion="4" />
</manifest>
```

Ниже приведено краткое описание строк кода, отмеченных номерами.

- ✓ **18.** Эта строка кода открывает элемент, регистрирующий объект `BroadcastReceiver` в приложении. Атрибут `name` данного элемента определяет имя приемника. В данном случае приемником служит объект `AppWidget`, определенный в файле `AppWidget.java`. В этом же атрибуте заданы действие и метка приемника.
- ✓ **21.** Идентификация типа намерения (на основе действия в фильтре намерений), на которое автоматически реагирует виджет приложения, когда намерение передается в широковещательный канал. Тип намерения называется *фильтром намерений* и помогает операционной системе Android определить, о каких событиях следует извещать ваше приложение. В данном случае приложение хочет получать действия `APPWIDGET_UPDATE` широковещательных намерений. Событие обновления генерируется свойством `updatePeriodMillis`, определенным в файле `widget_provider.xml`. Другие действия, в которых приложение может быть заинтересовано — включение, отключение, удаление и т.д.
- ✓ **25.** Идентификация размещения метаданных, встроенных в приложение. Операционная система Android использует метаданные для определения параметров приложения, установленных по умолчанию, и компоновки виджета приложения.

Сейчас все готово для установки виджета приложения на главный экран. Для этого выберите команду Run⇨Run (Выполнить⇨Выполнить) или нажмите клавиши <Ctrl+F11>. Приложение будет запущено в эмуляторе. Откройте в эмуляторе главный экран, щелкнув мышью на кнопке Home (Главный). Теперь можете добавить созданный вами виджет приложения на главный экран.

Добавление виджета на главный экран

Разработчики операционной системы Android поступили очень мудро, позволив размещать виджеты приложений на главном экране. Это существенно повысило удобство пользования устройствами Android и безопасность операционной системы. Разместить виджет на главном экране очень легко (конечно, когда приложение подготовлено к этому, чем мы, собственно, и занимались в данной главе). Для этого выполните следующие операции.

1. **Длительному прикосновению на сенсорном экране соответствует длительный щелчок мыши на экране эмулятора. Наведите указатель мыши на главный экран эмулятора, нажмите левую кнопку мыши и подержите несколько секунд.**
2. **На главном экране активизируется диалоговое окно Add to Home screen (Добавить на главный экран), показанное на рис. 7.5. Выберите команду Widgets.**
3. **В диалоговом окне Choose widget (Выбор виджета) найдите и выберите приложение Silent Mode Toggle (рис. 7.6).**

Рис. 7.5. Диалоговое окно добавления значков на главный экран

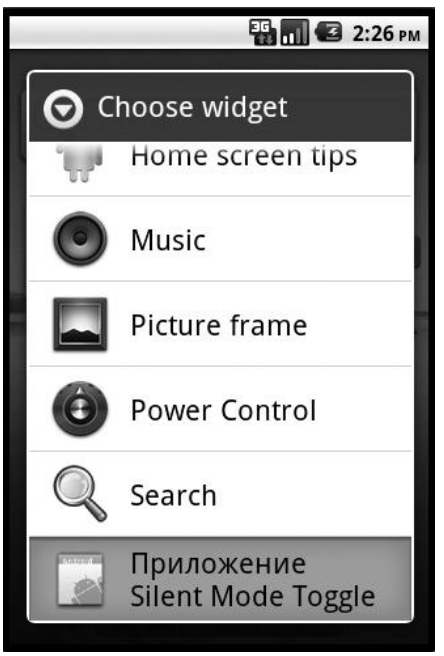


Рис. 7.6. Выбор добавляемого виджета

Вот и все! Значок виджета приложения Silent Mode Toggle должен появиться на главном экране (рис. 7.7). Прикоснитесь к этому значку (или щелкните на нем мышью), и вы увидите, что в результате будут выполнены две операции: изменится значок (см. рис. 7.2) и будет переключен режим звонка.

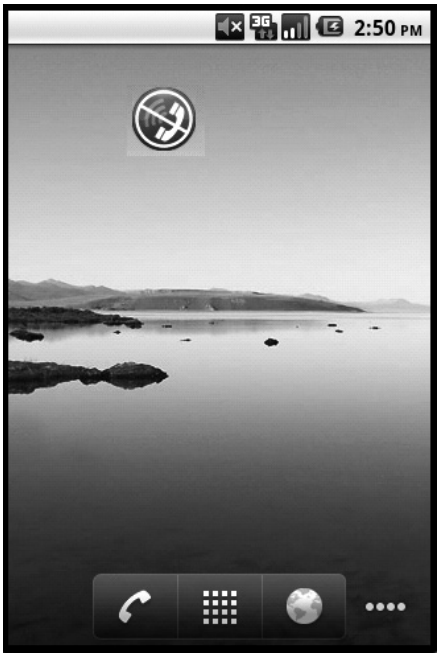


Рис. 7.7. Виджет приложения Silent Mode Toggle на главном экране эмулятора

Глава 8

Публикация приложения на сайте Android Market

В этой главе...

- Создание распространяемого файла
- Создание учетной записи Android Market
- Выбор правильной цены приложения
- Создание снимков экрана с вашим приложением
- Выгрузка приложения в Android Market
- Наблюдаем за количеством установленных экземпляров

Сайт Android Market представляет собой специальный механизм распространения приложений Android, поддерживаемый компанией Google. Если вы опубликуете свое приложение на Android Market, то миллионы пользователей во всем мире смогут загрузить, установить и использовать его. Кроме того, любой пользователь может выставить оценку вашему приложению, на основе которой вычисляется его рейтинг, и оставить свои комментарии на странице, посвященной приложению. Это поможет вам идентифицировать потенциальные тенденции популярности вашего приложения и обнаружить его слабые места.

Сайт Android Market предоставляет полезную статистику по опубликованному приложению (см. далее), на основе которой вы сможете отслеживать его успешность.

В данной главе рассматривается публикация приложения на сайте Android Market. Я предоставлю вам ряд снимков сайта, содержащих описание опубликованного приложения. Но чтобы опубликовать приложение, нужно упаковать его в распространяемый файл в специальном формате.

Создание распространяемого файла

Предположим, у вас появилась замечательная идея о создании для платформы Android чрезвычайно полезного приложения или очень интересной игры. Вы создали это приложение и готовы передать его в руки конечных пользователей. Как это сделать? В первую очередь, нужно упаковать приложение таким образом, чтобы его можно было развернуть на устройстве конечного пользователя. Для этого нужно создать файл APK (Android Package File — пакетный файл Android). В следующем разделе я проведу вас по этапам создания вашего первого файла APK.

Файл манифеста

Перед созданием распространяемого файла APK нужно сделать так, чтобы приложение было доступно как можно большему количеству пользователей. Для этого ознакомьтесь с элементом `uses-sdk` файла манифеста `AndroidManifest.xml`. В данный момент в приложении Silent Mode Toggle элемент `uses-sdk` содержит атрибут `minSdkVersion`, установленный при создании примера приложения в главе 4.

```
<uses-sdk android:minSdkVersion="4" />
```

Свойство `minSdkVersion` определяет минимальную версию платформы Android, на которой может быть установлено данное приложение. В данном случае установлена минимальная версия 4. Тем не менее при создании приложения Silent Mode Toggle в среде разработки Eclipse была установлена целевая версия 8. Что это означает? Какая разница между минимальной и целевой версиями?

Версии платформы Android почти всегда (за редкими исключениями) обладают свойством обратной совместимости. Например, почти все средства версии 3 есть в версии 4. Что означает здесь слово "почти". Лишь то, что платформа Android, как и все остальное в нашем мире, не идеальна. В такой сложной системе, обладающей тысячами разнообразных средств, невозможно уследить за всем и не совершить ни единой ошибки. В существующие средства иногда вносятся небольшие изменения, которые могут повлиять на предыдущие версии, хотя разработчики новой версии прикладывают огромные усилия, чтобы этого не происходило. К тому же, иногда в Android добавляются новые компоненты, которых нет в предыдущих версиях. Если в приложении применяется такой компонент, обратная совместимость, естественно, нарушается. Таким образом, если в приложении установлена минимальная версия 4, значит, оно будет работоспособным в устройстве, на котором установлена операционная система Android версии 4 и выше.

На основе значения атрибута `minSdkVersion` сайт Android Market определяет, какие приложения должны быть показаны пользователю конкретного устройства, поскольку пользователь указывает номер версии Android, установленной на его устройстве. Если в приложении установлено значение `minSdkVersion=4`, а пользователь укажет, что на его устройстве установлена версия 3 (т.е. Android 1.5) или более низкая, то пользователь не увидит данное приложение. Сайт Android Market отфильтрует все приложения версии 3 и ниже. Если же пользователь укажет версию 4 или выше, он увидит данное приложение и сможет установить его на своем устройстве.

Если в манифесте приложения в элементе `uses-sdk` опустить атрибут `minSdkVersion`, сайт Android Market автоматически подставит значение 0. Это означает, что данное приложение совместимо со всеми версиями Android. Если в приложении применяются компоненты, недоступные для старых версий платформы (например, средства Bluetooth, доступные только начиная с Android 2.0), устройство пользователя сгенерирует ошибку времени выполнения, сообщая, что выполнение приложения не может быть продолжено. Пользователь будет весьма недоволен тем, что потратил время и деньги на загрузку и установку ненужного приложения. Правда, экономически вы выигрываете, но такие действия считаются мошенничеством, причем это легко обнаружить.

Выбор наилучшего набора инструментов

Создать файл Android APK можно одним из следующих способов:

- ✓ с помощью надстройки ADT, встроенной в программу Eclipse;
- ✓ в автоматическом процессе сборки, встроенном в сервер непрерывной интеграции, такой как, например, Hudson Continuous Integration Server;
- ✓ в командной строке с помощью инструмента Ant;
- ✓ с помощью системы сборки Maven.

В данной книге для создания файла APK мы применим надстройку ADT программы Eclipse. Она предоставляет для этого ряд инструментов, которые компилируют, подписывают (создают цифровую подпись) и упаковывают приложение Android в единственный файл APK.

Другие варианты создания файла APK более сложные и применяются квалифицированными программистами только при необходимости, когда надстройка ADT не может предоставить желаемые средства. Дополнительную информацию о процессе сборки Ant и других инструментах создания файла APK можно найти в документации Android:

```
http://d.android.com/guide/publishing/app-signing.html
```

Цифровая подпись приложения

Операционная система Android требует, чтобы каждое установленное приложение имело цифровую подпись на основе сертификата, определяющего пару ключей (открытый и закрытый). Закрытый ключ известен только разработчику. Сертификат, примененный для создания цифровой подписи, используется для идентификации приложения и установки доверительных отношений между приложениями.

Чтобы правильно подписать приложение Android, необходимо знать следующее.

- ✓ Все приложения Android должны быть подписанными. Операционная система не установит неподписанное приложение.
- ✓ Для создания цифровой подписи можете использовать собственный сертификат, никакие официальные органы сертифицирования для этого не нужны.
- ✓ Когда приложение готово к поставке на рынок, нужно подписать его с помощью закрытого ключа. Нельзя подписать приложение с отладочным ключом, который подписывал файл APK в процессе разработки приложения.
- ✓ Сертификат имеет конечную дату, которая проверяется только в момент установки приложения. Если конечная дата наступит после установки приложения на устройстве, приложение будет продолжать нормально работать.
- ✓ Если для генерации сертификата использовать инструменты ADT по какой-либо причине нежелательно, для создания и подписания файла APK можно применить любой стандартный инструмент, например Keytool или Jarsigner.

Можно создать приложения, сообщающиеся друг с другом. Они должны быть подписаны с помощью одного и того же ключа. Это позволяет приложениям выполняться в одном процессе. При запросе система считает их одним приложением. Данная методика позволяет создать приложение, состоящее из модулей. Пользователи могут обновлять каждый модуль отдельно от других по мере необходимости, решая, есть ли смысл покупать очередное обновление данного модуля.

Процедура сертификации подробно рассматривается в документации Android. В ней описано, как генерировать сертификаты с помощью разных инструментов и методик. Дополнительную информацию о подписании файлов APK можно найти по такому адресу:

```
http://d.android.com/guide/publishing/app-signing.html
```

Создание хранилища ключей

Хранилище ключей в Android (как и в Java) представляет собой контейнер, в котором размещены ваши персональные сертификаты. Создать файл хранилища ключей можно с помощью следующих инструментов.

- ✓ **Мастер экспорта ADT.** Этот инструмент встроен в настройку ADT и позволяет экспортировать подписанные файлы APK, а также создавать сертификаты и хранилища ключей в пошаговом процессе мастера.
- ✓ **Приложение Keytool.** Позволяет создать хранилище ключей с помощью командной строки. Этот инструмент можно найти в папке `tools` пакета Android SDK. Он предоставляет в командной строке много полезных параметров сертификации.

В данной книге для создания хранилища ключей и генерации файла APK применяется мастер экспорта ADT.

Безопасность хранилища ключей

В файле хранилища ключей находится ваш закрытый сертификат, используемый платформой Android для идентификации вашего приложения на сайте Android Market. Создайте резервную копию хранилища ключей в безопасном месте. Если вы потеряете его, то не сможете больше подписывать свои приложения тем же закрытым ключом. Следовательно, вы не сможете обновлять свое приложение, потому что сайт Android Market увидит, что ваше приложение подписано другим ключом и не позволит обновить его. В этом случае сайт "видит" файл обновления как новое, другое приложение Android. Это же произойдет, если изменить имя пакета приложения. В таком случае Android Market тоже не позволит обновить приложение (см. далее).

Создание файла APK

Чтобы создать файл APK с помощью надстройки ADT, выполните следующие операции.

1. **Откройте программу Eclipse.**
2. **Щелкните правой кнопкой мыши на приложении Silent Mode Toggle и выберите в контекстном меню команду Android Tools⇨Export Signed Application Pacage (Инструменты Android⇨Экспортировать в пакет подписанного приложения).**

Активизируется диалоговое окно экспорта приложения (рис. 8.1) с подставленным именем текущего проекта.

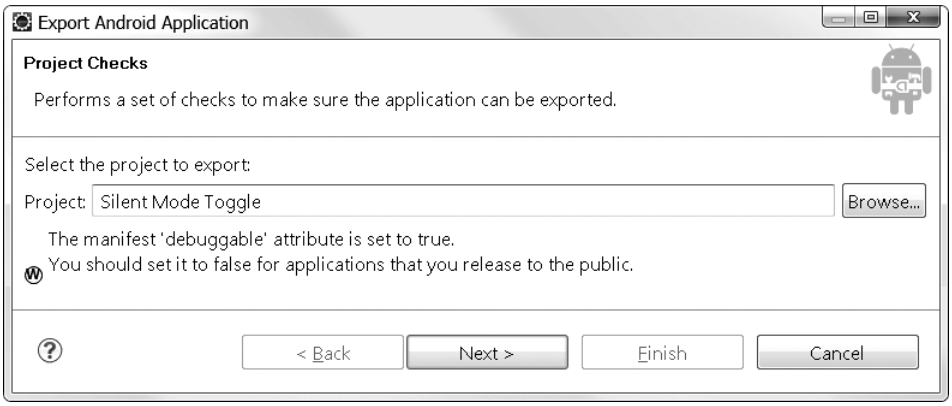


Рис. 8.1. Первое окно мастера экспорта приложения

3. **Щелкните на кнопке Next (Далее).**
   Активизируется диалоговое окно Keystore Selection (Выбор хранилища ключей), показанное на рис. 8.2.

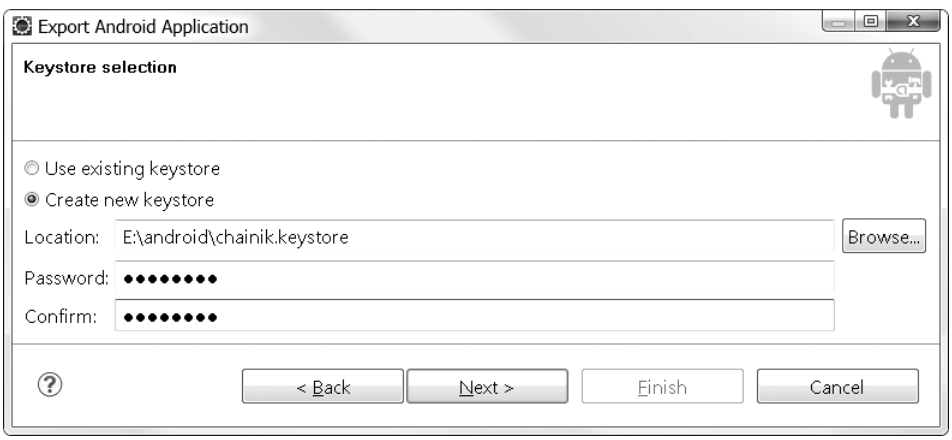


Рис. 8.2. Создание хранилища ключей

4. **Пока что вы еще не создали хранилище ключей, поэтому установите переключатель Create new keystore (Создать хранилище ключей).**
5. **В текстовом поле Location (Местоположение) введите или выберите адрес хранилища.**
   Рекомендуется поместить хранилище в папку Android по адресу `E:\android`. Имя файла должно иметь расширение `.keystore`. Полностью адрес файла выглядит следующим образом:

```
E:\android\chainik.keystore
```

6. **Придумайте и введите пароль, который следует запомнить. Повторно введите пароль в поле Confirm (Подтвердить).**
7. **Щелкните на кнопке Next.**

   Активизируется диалоговое окно Key Creation (Создание ключей).
8. **Заполните следующие поля.**
   - Alias (Псевдоним). Используется для визуальной идентификации ключа.
   - Password (Пароль) и Confirm (Подтверждение). Этот пароль вы будете вводить при использовании ключа.
   - Validity (Срок действительности). Значение, заданное в этом поле, определяет, как долго можно будет пользоваться ключом. В данном примере задайте достаточно большой интервал времени.
9. **Заполните следующие поля диалогового окна (не обязательно все, заполнить нужно как минимум одно поле):**
   - First and Last Name (Имя и фамилия);
   - Organization Unit (Подразделение);
   - Organization (Организация);
   - City or Locality (Город или населенный пункт);
   - State or Province (Штат или провинция);
   - Country Code (Код страны).

   Диалоговое окно должно выглядеть, как показано на рис. 8.3.

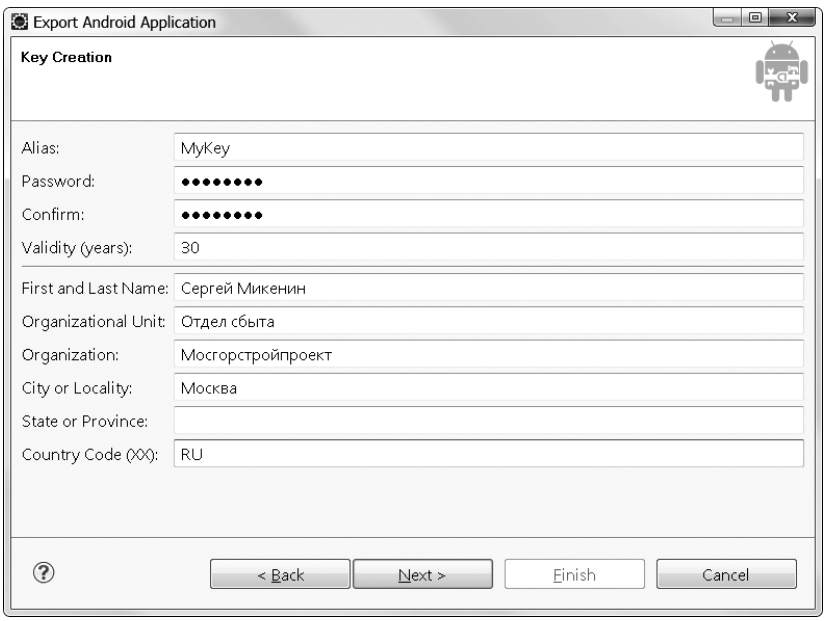


Рис. 8.3. Ввод информации о ключе

10. **Щелкните на кнопке Next.**

    Активизируется последнее окно мастера экспорта (рис. 8.4).

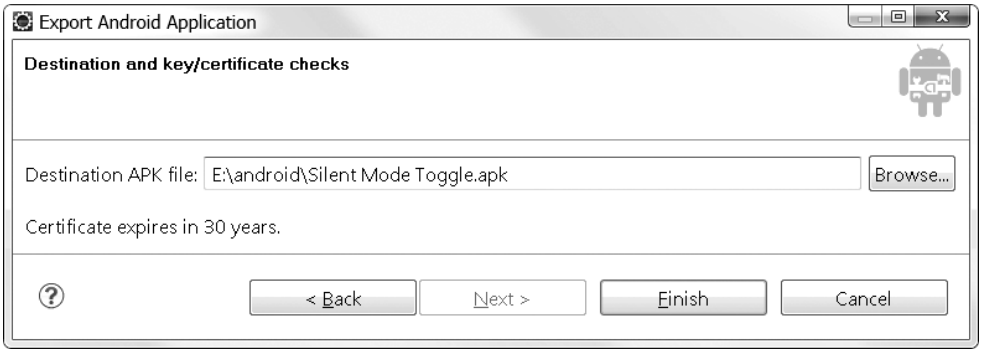


Рис. 8.4. Задание имени и маршрута файла APK

11. **Введите имя и маршрут файла с расширением `.apk`.**
12. **Щелкните на кнопке Finish (Готово).**

В заданной папке будут созданы файлы `.apk` и `.keystore` (рис. 8.5).

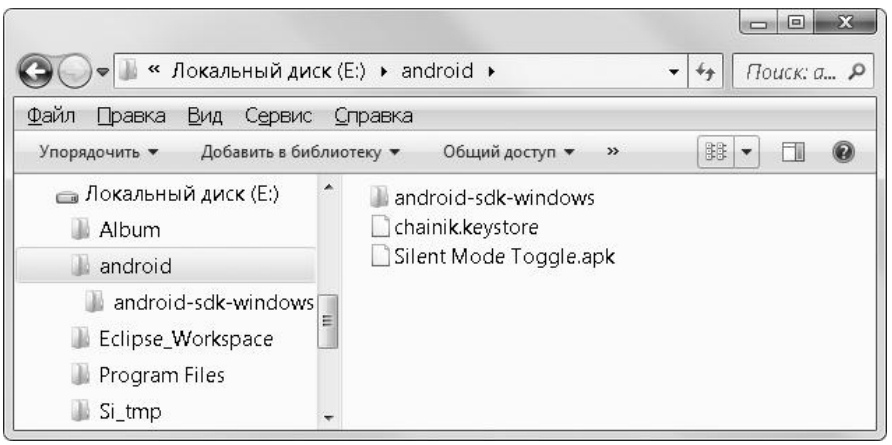


Рис. 8.5. Файлы `.apk` и `.keystore` в файловой системе компьютера

Создание учетной записи Android Market

Теперь, когда у вас есть подписанный файл APK, можете опубликовать приложение на сайте Android Market. Для этого нужно создать свою учетную запись Android Market. Но чтобы создать ее, нужно иметь учетную запись Google. Подойдет любая учетная запись Google, например Gmail. Если у вас нет учетной записи Google, можете получить ее бесплатно на странице `http://www.google.com/accounts`. После этого для создания учетной записи Android Market выполните приведенные ниже операции. Обратите внимание на то, что вам придется внести плату за регистрацию в размере 25 долларов США с помощью платежной карточки. Если не оплатите регистрацию, то не сможете опубликовать приложение.

1. **Запустите веб-браузер и откройте страницу `http://market.android.com/publish`.**
2. **Зарегистрируйтесь в своей учетной записи Google (рис. 8.6).**

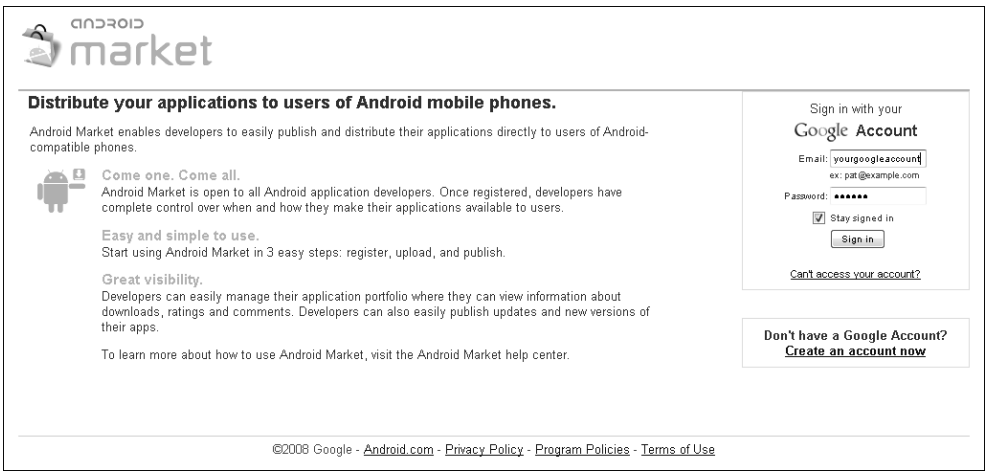


Рис. 8.6. Страница регистрации

3. **На следующей странице заполните следующие поля.**
   - **Developer Name** (Имя разработчика). Ваши имя и фамилия (можете использовать свой псевдоним), которые на сайте будут указаны как имя и фамилия разработчика опубликованного приложения. В это поле можно ввести название компании. Содержимое поля Developer Name можно будет изменить после создания учетной записи.
   - **Email Address** (Электронный адрес). Адрес, по которому пользователи будут отправлять вам письма. Обычно пользователи задают вопросы или комментируют качество приложения.
   - **Website URL** (URL веб-сайта). Адрес вашего веб-сайта. Если у вас нет веб-сайта, можете получить бесплатную учетную запись Blogger, предоставляющую бесплатный блог. Сайт Android Market будет считать этот блог вашим веб-сайтом. Бесплатную учетную запись Blogger можно получить по адресу `www.blogger.com`.
   - **Phone Number** (Номер телефона). Номер, по которому с вами можно связаться в случае возникновения проблем с опубликованным содержимым.

   После заполнения форма должна выглядеть приблизительно так, как на рис. 8.7.
4. **Щелкните на кнопке Continue (Продолжить).**

   На следующей странице отображено приглашение внести плату за регистрацию в размере 25 долларов США (рис. 8.8).
5. **Щелкните на кнопке Continue (Продолжить), чтобы оплатить регистрацию с помощью учетной записи Google Checkout.**

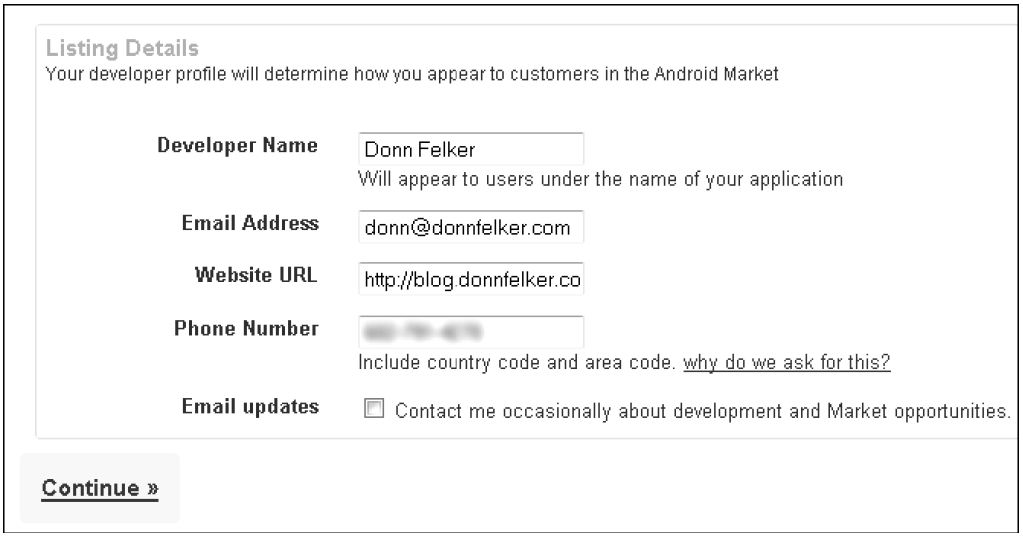


Рис. 8.7. Информация о разработчике приложения

Рис. 8.8. Плата за регистрацию

6. **На странице, показанной на рис. 8.9, введите параметры вашей кредитной карточки и другую платежную информацию. Щелкните на кнопке Agree and Contingue (Подтвердить и продолжить).**

   Если ваша кредитная карточка уже зарегистрирована в Google, вы не увидите эту страницу. В этом случае на появившейся странице выберите свою карточку и щелкните на кнопке Continue.
7. **На следующей странице (рис. 8.10) введите пароль и щелкните на кнопке Sign in and continue (Подписаться и продолжить).**

Рис. 8.9. Информация о кредитной карточке

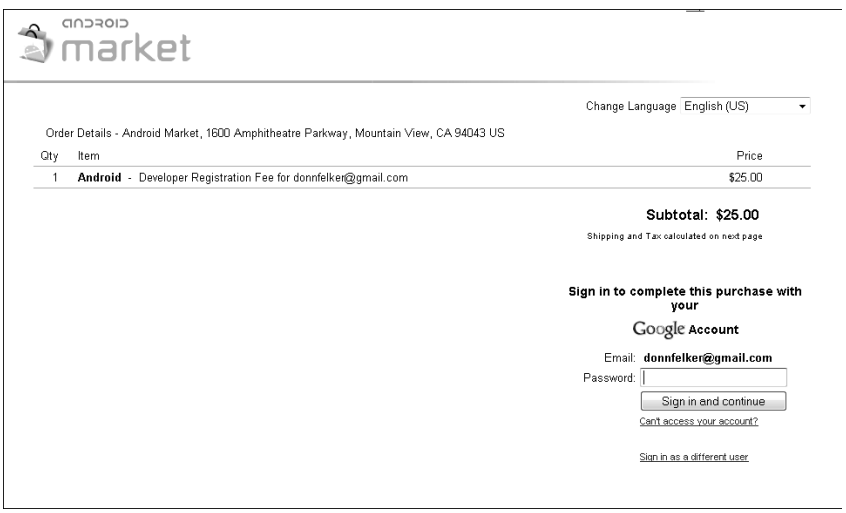


Рис. 8.10. Подтверждение подписки для регистрации в качестве разработчика

8. **На странице подтверждения заказа регистрации (рис. 8.11) щелкните на кнопке Place your order now (Сделать заказ сейчас).**

   В зависимости от быстродействия вашего соединения с Интернетом и загруженности сайта в данный момент, вы можете не увидеть загрузочной страницы. Когда процесс будет завершен, вы увидите сообщение, подтверждающее, что вы зарегистрированы как разработчик приложений Android (рис. 8.12).

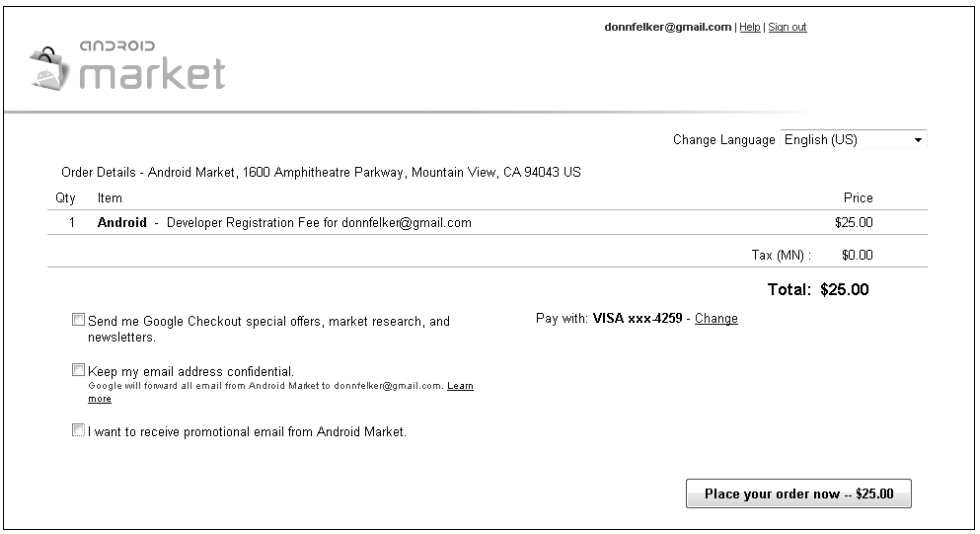


Рис. 8.11. Подтвердите требование зарегистрировать вас как разработчика

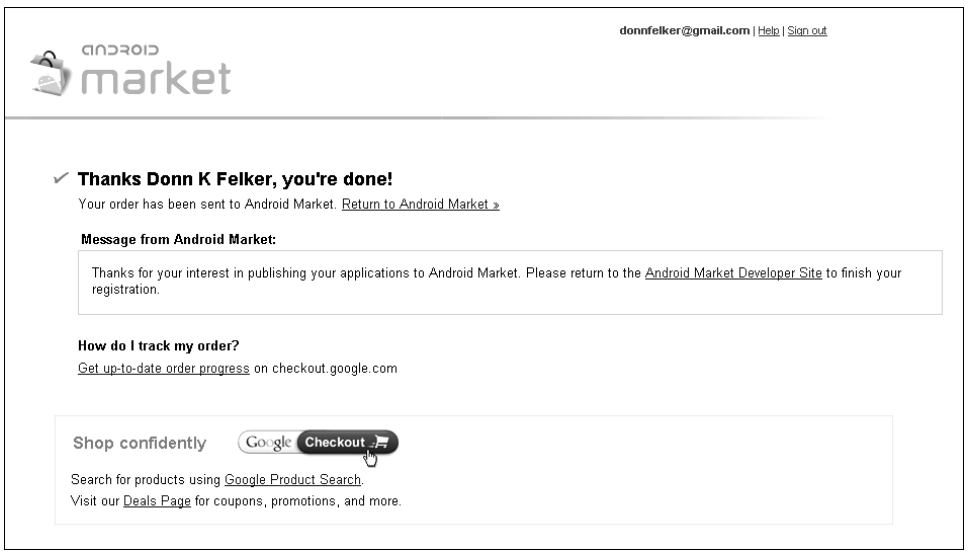


Рис. 8.12. Сообщение о том, что вы зарегистрированы как разработчик

9. **Щелкните на гиперссылке `Android Market Developer Site` (Сайт разработчика Android Market).**

   Откроется страница подтверждения соглашений между сайтом и разработчиком (рис. 8.13).

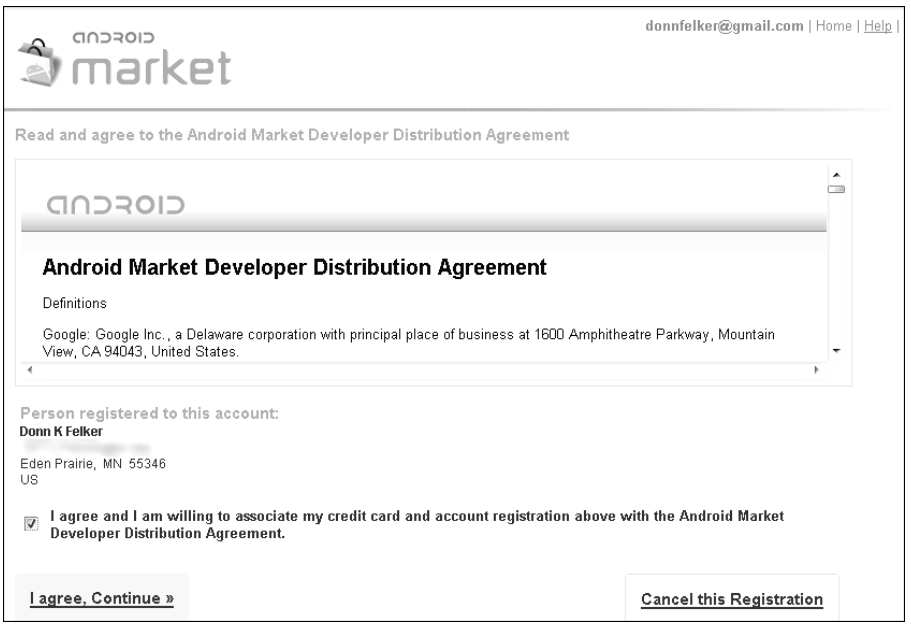


Рис. 8.13. Страница подтверждения соглашений

10. **Если вы планируете разместить на сайте Android Market платное приложение и получать за него деньги, прочитайте врезку о платежной учетной записи Google.**
11. **На странице подтверждения соглашений прочитайте условия соглашений, установите флажок I agree... (Я согласен...) и щелкните на гиперссылке `I agree, Continue` (Я согласен, продолжить).**

    Будет открыта начальная страница разработчика приложений Android (рис. 8.14).

Платежная учетная запись Google Checkout

Чтобы получать деньги за приложение, размещенное на сайте Android Market, нужно создать и сконфигурировать свою платежную учетную запись Google Checkout. Для этого щелкните на гиперссылке **`Setup Merchant Account`** (Создать платежную учетную запись) и введите следующие данные:

- ✓ личная и деловая информация о себе;
- ✓ налоговая информация (личная или корпоративная);
- ✓ ожидаемый ежемесячный доход.

После создания платежной учетной записи Google Checkout вы сможете продавать свое приложение. Для этого вернитесь к п. 11 предыдущей инструкции, чтобы выгрузить свое приложение на сайт. Подробнее этот процесс рассматривается далее.

Рис. 8.14. Начальная страница разработчика

Выбор правильной цены приложения

Теперь у вас есть файл APK, и вы являетесь зарегистрированным разработчиком Android. Вы готовы передать созданное вами приложение в руки пользователей. Но осталось решить еще одну, последнюю проблему. Как предоставить приложение пользователям: бесплатно или за деньги?

Это решение нужно принять перед публикацией приложения, потому что оно имеет психологические последствия для потенциальных пользователей и экономические последствия для вас. Если решите сделать приложение платным, назначьте ему оптимальную цену. Никто (и я в том числе) не сможет выбрать цену за вас, я могу лишь дать несколько советов, как человек более опытный в этом деле. Просмотрите аналогичные приложения на сайте Android Market. Ознакомление с ними и, конечно же, с их ценами поможет вам определить свою ценовую политику. На сайте Android Market вы увидите, что большинство приложений попадает в ценовой диапазон от 0,99 до 9,99 долларов. Приложения дороже 10 долларов встречаются довольно редко. Цена вашего продукта должна быть конкурентоспособной на фоне других, аналогичных продуктов. Если ваш продукт уникален, можете назначить более высокую цену, но тогда объясните потенциальным пользователям, чем он уникален. Ценовая политика — это игра, в которую вы должны включиться, чтобы найти оптимальную цену для вашего приложения.

Преимущества и недостатки платного и бесплатного распространения приложений — тема непрекращающихся дискуссий на многочисленных сайтах и форумах. Сторонники обеих точек зрения согласны с тем, что оба подхода могут приносить прибыль. Как разработчик, я применял оба подхода и на основании своего опыта обнаружил, что оба могут быть весьма прибыльными. Вы должны лишь выяснить, какой подход лучше для вашего приложения в конкретной ситуации.

Преимущества платной модели

Выбор платной модели означает, что деньги появятся в вашем кармане немедленно после первой продажи (в течение 24 часов, исключая выходные и праздники; в этом случае вы получите свои деньги в первый рабочий день). Однако мой опыт свидетельствует, что платное приложение загружают и устанавливают намного меньше пользователей, чем бесплатное. Фактически вы берете на себя функции рекламодателя своего собственного приложения. О нем никто ничего не знает. Привлечь внимание к приложению — непростая задача. Аналогичная проблема возникает и с бесплатным приложением, но в этом случае пользователь может бесплатно и легко установить его на своем устройстве и ознакомиться с ним. В то же время, ознакомление с платным приложением существенно ограничено.

Все пользователи Android Market могут 24 часа бесплатно пользоваться любым платным приложением. Вы, как пользователь, инициируете предварительную покупку приложения и устанавливаете его на своем физическом устройстве. Ваша учетная запись Google Checkout авторизует платежную карточку и оплата остается в состоянии авторизации на протяжении 24 часов с момента предварительной покупки. Вы можете наблюдать состояние учетной записи на панели Google Checkout. На протяжении 24 часов вы имеете в своем распоряжении полнофункциональное приложение. Если оно вам не понравилось, вы деинсталлируете его и отменяете процедуру авторизации. Если в течение 24 часов вы не деинсталлируете приложение, процедура авторизации автоматически пришлет вам счет, который вы должны оплатить. Психологически пользователь чувствует себя очень комфортно, когда знает, что ему не придется платить за то, что ему не понравилось.

Преимущества бесплатной модели

Если вы решите воспользоваться бесплатной моделью, пользователи смогут загружать и устанавливать ваше приложение, не заплатив ни копейки. Согласно статистике, 50–80% пользователей, которые установили бесплатное приложение, оставляют его на своем устройстве, а остальные удаляют его. Однако как вы на этом заработаете деньги? В чем ваша выгода?

Ответ прост: за счет рекламы. Существует множество рекламных агентств, работающих с мобильными устройствами и предоставляющих разработчикам библиотеки сторонних поставщиков. Библиотеки отображают рекламу в мобильных приложениях. Наиболее известные рекламные компании, занимающиеся мобильными устройствами, — Google AdSense, AdMob (недавно приобретена Google) и Quattro Wireless (недавно приобретена Apple). Получить бесплатную учетную запись в любой из этих компаний очень легко. Они предоставляют пакет инструментов для встраивания рекламных компонентов в приложение Android. Традиционно большинство рекламных компаний выплачивает гонорар регулярно (каждые 60 дней), поэтому, прежде чем вы получите свой первый чек, может пройти два месяца.

Создание снимков экрана с вашим приложением

Снимки экрана — важная часть инфраструктуры Android Market, потому что они позволяют потенциальным пользователям увидеть ваше приложение до его установки

в мобильное устройство. Несколько снимков экрана выполняющегося приложения — решающий фактор, влияющий на то, установит ли пользователь ваше приложение. Предположим, вы создали игру и хотите, чтобы пользователи играли в нее. Вы потратили недели или даже месяцы на создание привлекательной графики и, естественно, хотите, чтобы потенциальные покупатели увидели и оценили ее. Для этого нужно создать снимок экрана во время работы приложения.

Чтобы извлечь снимок экрана вашего приложения в режиме реального времени, необходим эмулятор или физическое устройство Android. Выполните следующие операции.

1. **Откройте эмулятор и поместите виджет на главный экран.**
2. **В Eclipse откройте окно DDMS.**
3. **Выберите эмулятор (если он у вас не один) на панели Devices (Устройства), как показано на рис. 8.15.**
4. **Щелкните на кнопке Screen Shot (Снимок экрана), чтобы сгенерировать снимок.**
5. **Щелкните на кнопке Save (Сохранить) и сохраните снимок в подходящей для него папке.**

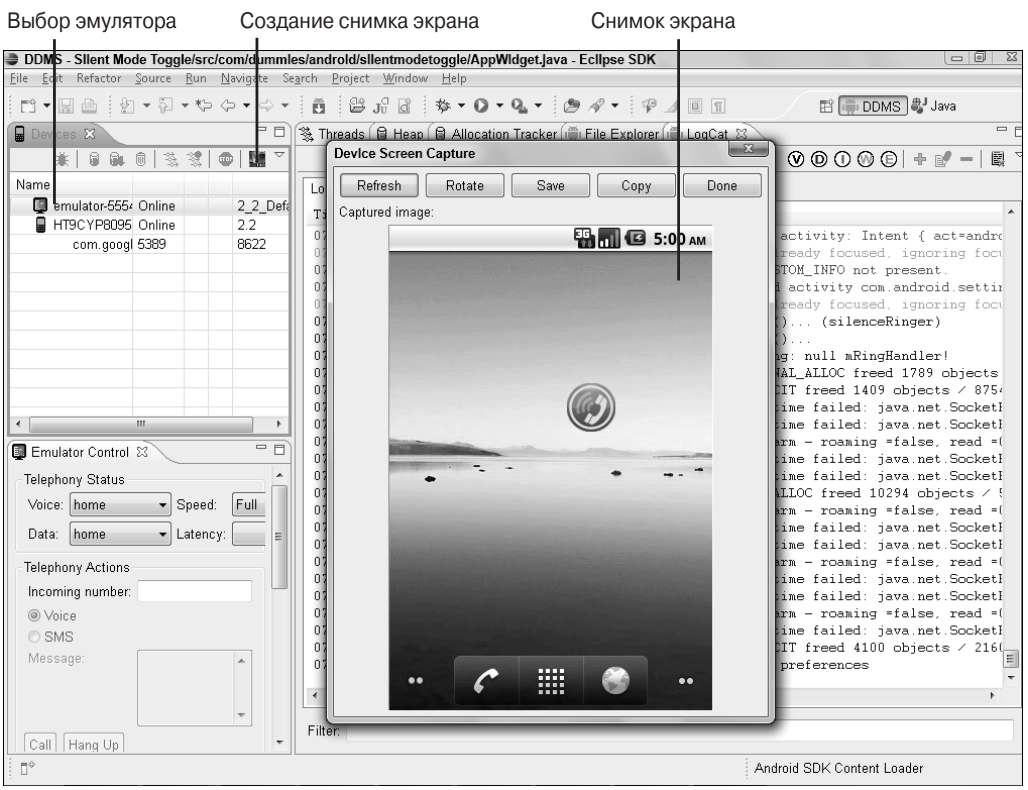


Рис. 8.15. Окно DDMS и снимок экрана эмулятора

Выгрузка приложения в Android Market

Итак, вы достигли завершающего этапа создания приложения Android — момента его представления всему миру. Чтобы опубликовать приложение, выполните следующие действия.

1. **На странице разработчика Android (см. рис. 8.14) щелкните на кнопке Upload Application (Выгрузить приложение).**

   Активизируется страница выгрузки (рис. 8.16).

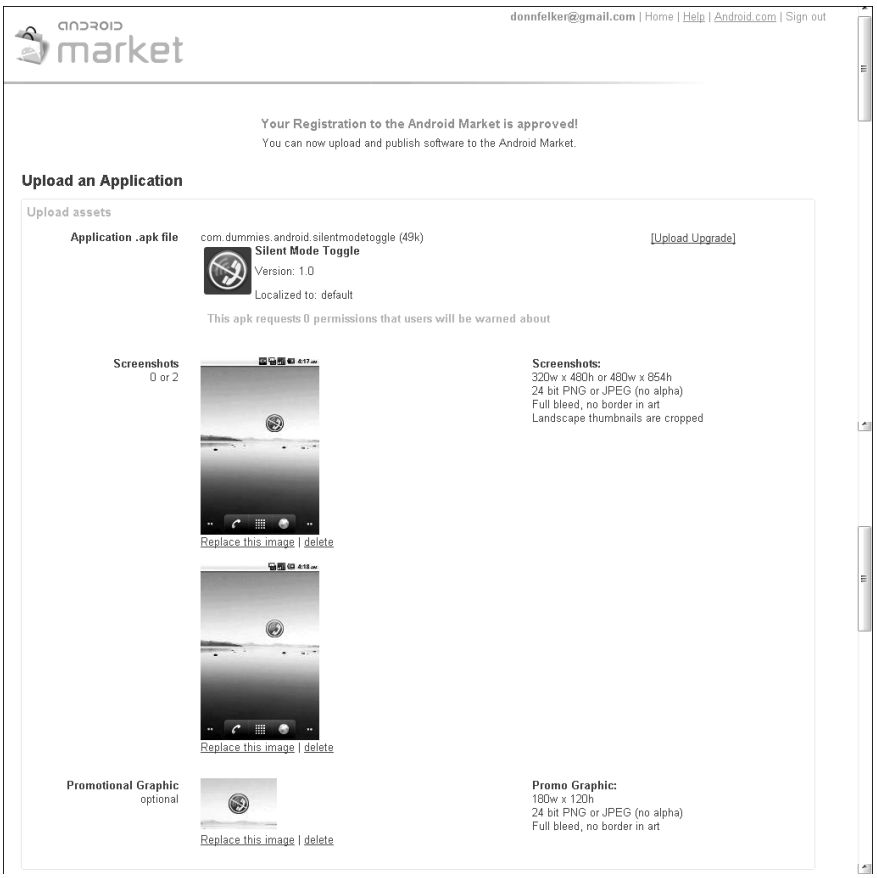


Рис. 8.16. Страница выгрузки приложения на сервер Android Market

2. **В разделе Application .apk file (Файл .apk приложения) выберите файл APK, созданный вами ранее, и щелкните на кнопке Upload (Выгрузить).**

На сайте Android Market имя пакета Java используется в качестве идентификатора приложения. Поэтому никакие два приложения не могут носить одно и то же имя пакета. Приложение Silent Mode Toggle носит имя пакета `com.dummies.android.silentmodetoggle`. Работая над книгой, я выгрузил данный пакет на сайт Android Market, поэтому, если вы попытаетесь для

эксперимента выгрузить приложение Silent Mode Toggle, не изменив имя пакета, вы получите сообщение о том, что выгрузка невозможна, потому что такой пакет существует на сайте у другого разработчика. Чтобы имя пакета было уникальным, введите в него ваше имя или название вашей организации.

3. **В разделе Screenshots (Снимки экрана) добавьте два снимка вашего приложения.**

   Снимки должны иметь размеры 320×480 или 480×854 пикселей. Благодаря снимкам пользователь может увидеть приложение в рабочем состоянии до его установки на физическом устройстве. В принципе, снимки экрана можно не предоставлять, но статистика свидетельствует, что пользователи устанавливают приложения, сопровождаемые снимками, гораздо чаще, чем приложения без снимков. Для публикации снимки приложения не обязательны.

4. **Добавьте рекламный снимок экрана.**

   Рекламный снимок должен иметь размеры 180×120 пикселей. Его нужно создать в графической программе, позволяющей редактировать файлы изображений. Рекламный снимок используется сайтом Android Market во многих местах для продвижения вашего приложения. Для публикации приложения рекламный снимок не обязателен.

5. **Введите название приложения.**

   Для рассматриваемого в данной главе приложения выбрано название Silent Mode Toggle Widget. Текст названия индексируется в поисковой системе Android Market. Следовательно, количество пользователей, которые найдут ваше приложение, существенно зависит от того, насколько удачное название вы для него выбрали.

6. **Добавьте описание приложения.**

   Это описание пользователи увидят, когда откроют страницу Android Market, посвященную вашему приложению, и будут решать, устанавливать его или нет. Весь текст описания индексируется поисковой системой Android Market.

7. **Добавьте рекламный текст приложения.**

   Рекламный текст используется для оценки и продвижения вашего приложения сайтом Android Market. Процесс оценки приложения довольно сложный, но очевидно, что основной фактор — популярность приложения. Если приложение выбрано сайтом для оценивания и продвижения, оно отображается в области продвижения, которая обычно расположена в верхней части экрана каждой категории Android Market. Рекламный текст отображается как компонент информации, предназначенной для продвижения оцениваемого приложения.

8. **Задайте тип приложения.**

   В данном случае я установил тип Application (Приложение).

9. **Задайте категорию приложения.**

   Для Silent Mode Toggle я выбрал категорию Productivity (Производительность), потому что приложение предназначено для повышения производительности устройства.

10. **Выберите способ защиты установленного экземпляра приложения (copy protection).**

Я всегда выбираю значение Off (Отключить). При значении On (Включить) в устройстве для приложения резервируется удвоенный объем памяти. Например, если приложение имеет размер 2 Мбайт, для него резервируется приблизительно 4 Мбайт. Приложению желательно всегда занимать как можно меньше места, потому что при нехватке памяти в мобильном устройстве пользователи удаляют в первую очередь крупные приложения, чтобы освободить как можно больше места.

В предыдущих версиях (до Android 2.2) приложения нельзя было устанавливать на карту SD. Внутренняя память мобильного устройства обычно очень ограниченная, и когда пользователь исчерпывает ее, он удаляет крупные приложения. Поддерживая малые размеры файла APK и отключая защиту установленного экземпляра приложения, вы увеличиваете его шансы остаться в мобильном устройстве пользователя.

11. **Создайте список мест, в которых ваше приложение должно быть видимым.**

    Например, если в приложении применяется итальянский язык и оно предназначено для Италии, снимите флажок All locations (Все места) и установите флажок Italy (Италия). Тогда на сайте Android Market приложение увидят только итальянские пользователи. Если оставить флажок All locations, приложение будет видимым везде.

12. **Заполните поля Web Site (Веб-сайт) и E-mail (Электронная почта). При желании можете заполнить также поле Phone (Телефон).**

    Я никогда не заполняю поле Phone, потому что пользователь может позвонить с другого континента посреди ночи и задать какой-нибудь вопрос. Я предпочитаю общаться с пользователями посредством электронной почты. Если вы создаете приложение для какой-либо компании, а публикуете его со своей учетной записью разработчика, можете задать номер телефона компании, чтобы пользователи не тревожили вас. Поля Web Site и E-mail позволяют получать от пользователей сообщения об ошибках и пожелания относительно добавления в приложение новых средств.

13. **Проверьте, не нарушены ли требования Android Market к содержимому приложений. Установите все флажки, удостоверяющие ваши знакомство и согласие с действующим законодательством в данной области.**

14. **Щелкните на одной из следующих кнопок.**

    - **Publish** (Опубликовать). Сохранение и публикация приложения на сайте Android Market.
    - **Save** (Сохранить). Сайт сохраняет внесенные изменения, но не публикует приложение.
    - **Delete** (Удалить). Удаление приложения.

    В данном упражнении щелкните на кнопке Save. Приложение будет сохранено, и вы вернетесь на главную страницу разработчика Android. На странице вы увидите отметку Saved Draft (Сохраненный черновик), извещающую о том, что приложение находится в сохраненном состоянии (рис. 8.17). Можете считать это состояние тестовым (удостоверяющим, что вы сделали все правильно), пока не будете готовы к поставке отлаженного приложения.

Рис. 8.17. Сохраненное приложение на главной странице разработчика

15. **Когда приложение будет готово к поставке, щелкните на названии приложения на главной странице разработчика Android.**

    Активизируется страница выгрузки приложения (см. рис. 8.16).

16. **Прокрутите страницу вниз, чтобы была видна кнопка Publish (Опубликовать), и щелкните на ней.**

    Приложение будет опубликовано на сайте Android Market.

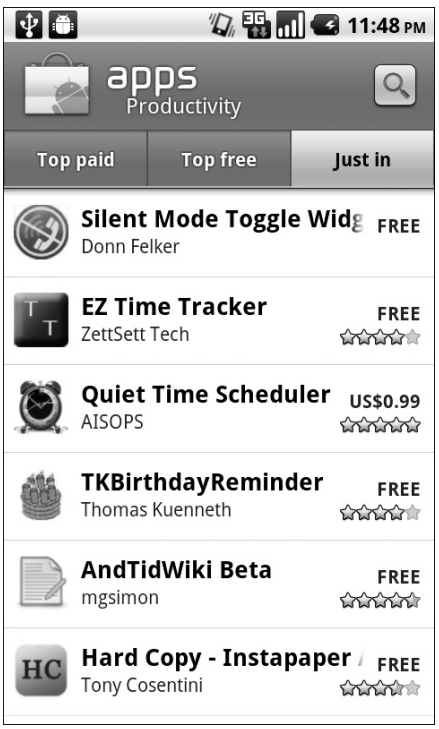


Рис. 8.18. Приложения, опубликованные на сайте Android Market

На рис. 8.18 показано приложение Silent Mode Toggle Widget, опубликованное на сайте Android Market и отображаемое на мобильном устройстве Nexus One. Чтобы увидеть его, я открыл сайт Android Market, открыл категорию Apps — Productivity (Приложения — производительность), которую я назначил для данного приложения, и активизировал вкладку Just in (Только что добавленные), в которой расположены недавно опубликованные приложения.

Обратите внимание на существенное преимущество процесса публикации по сравнению с другими мобильными платформами, состоящее в том, что процедура утверждения публикации отсутствует полностью. Вы можете создать любое приложение и опубликовать его прямо сейчас. После этого пользователи смогут устанавливать его в своих мобильных устройствах. Это означает, что вы можете сократить до минимума цикл поставки новых средств и версий приложения. Продолжительность цикла поставки зависит только от вас.

Если вы попытаетесь найти приложение Silent Mode Toggle Widget на сайте Android Market, то вам это не удастся, потому что я удалил его после написания книги. Данное приложение предназначено для демонстрации процесса разработки и публикации, а не для распространения среди реальных пользователей. Чтобы удалить приложение, я выбрал его название на главной странице разработчика Android, прокрутил открывшуюся страницу вниз и щелкнул на кнопке Unpublish (Отменить публикацию).

Наблюдаем за количеством установленных экземпляров

Итак, вы создали и опубликовали свое первое приложение. Теперь можете расслабиться, усесться в кресле поудобнее и наблюдать, как увеличивается количество пользователей, загрузивших и установивших ваше приложение. Независимо от того, кто вы — независимый программист, создающий первоклассные "стрелялки", или работник корпорации, публикующий приложения Android вашей компании, — вам не безразлично, как используется ваше приложение, нравится ли оно пользователям и устанавливают ли его пользователи на своих мобильных устройствах. Впрочем, количество установленных экземпляров приложения — не единственный и даже не самый важный показатель его успеха. Ниже перечислены наиболее важные показатели успешности приложения, которые можно наблюдать на сайте Android Market.

- ✓ **Рейтинг на основе пяти звездочек.** Каждый пользователь может выставить оценку вашему приложению. На основе оценок пользователей сайт вычисляет рейтинг вашего приложения. Естественно, чем выше рейтинг, тем лучше.
- ✓ **Комментарии.** Обязательно читайте их! Люди тратят время на их написание, поэтому вы должны прочитать их даже просто из вежливости. К тому же, вы будете удивлены тем, как много ценных идей пользователи предоставляют вам бесплатно. На основе собственного опыта я убедился в том, что реализация большинства запрошенных пользователями новых средств приводит к тому, что пользователи бурно реагируют на них, благодарят за понимание и устанавливают новые версии приложения.
- ✓ **Отчеты об ошибках.** Пользователи, достаточно любезные для того, чтобы предоставить вам отчеты об ошибках, хотят, чтобы вы знали, что во время выполнения приложение генерирует исключения по неизвестным причинам. Откройте эти отчеты, посмотрите на ошибки, отследите стек вызовов и попытайтесь исправить каждую ошибку. Если приложение часто терпит крах, вы быстро получите большое количество гневных отзывов. Учитывайте также, что стеки вызовов доступны только в устройствах Android 2.2 и выше.
- ✓ **Количество инсталляций приложения и количество работающих приложений.** То, что пользователь установил ваше приложение, еще не значит, что оно ему нравится. Возможно, он через некоторое время удалит его. Поэтому количество инсталляций — ненадежный показатель. Значительно лучший показатель — количество экземпляров вашего приложения, работающих на устройствах пользователей.

- ✓ **Электронные письма.** Поработав с приложением, пользователи будут возвращаться на Android Market, чтобы найти адреса ваших электронной почты и веб-сайта. Они будут посылать вам электронные письма с вопросами о средствах приложения и комментариями о том, почему им понравилось или не понравилось приложение. Часто пользователи передают в электронных письмах ценные идеи об улучшении приложения или просьбы создать приложение, которого еще нет на Android Market. Людям нравится быть в первых рядах. Им очень нравится, когда их мысли находят отклик. Я заметил, что если ответить на запросы в течение 24 часов, пользователи стремятся к конструктивному сотрудничеству. Если есть возможность, я всегда стараюсь ответить в течение 4 часов. Конечно, если ваше приложение установили миллионы пользователей, то вы не сможете ответить на каждое письмо, но тем не менее, устанавливая приложение, пользователи чувствуют себя намного комфортнее, когда знают, что при возникновении проблем всегда могут связаться с разработчиком.

Поддержка контакта с пользователями — весьма тяжелая работа, но она всегда окупается сторицей, потому что благожелательные и довольные пользователи будут рассказывать своим друзьям и родственникам о вашем приложении, расширяя тем самым круг его пользователей.

В 2011 году компания Amazon по примеру Google открыла сайт App Store, аналогичный сайту Android Market. С помощью этого сайта пользователи могут просматривать, покупать, загружать и устанавливать приложения. Разработчики могут продавать свои приложения через App Store, получать их рейтинги, генерируемые Amazon, и размещать бесплатные приложения. Кроме того, Amazon предоставляет отличные показатели продаж для разработчиков и специалистов по маркетингу. Дополнительную информацию можно найти по адресу `http://developer.amazon.com`.

Часть III

Создание мощных приложений

The 5th Wave

Рич Теннант

В этой части...

В части III мы расширим знания об Android, приобретенные в части II, перейдя к созданию мощных приложений. В этой части я не буду подробно объяснять каждую деталь, как в части II, однако приведу все сведения, необходимые для разработки мощных приложений Android. Кроме того, я попытаюсь перекинуть мостик через пропасть, отделяющую новичков от профессиональных разработчиков.

В данной части вы узнаете, как и зачем используются средства, увеличивающие полезность приложения. В этой части вы создадите полнофункциональное мощное приложение, взаимодействующее с локальной базой данных и предоставляющее средства настройки предпочтений пользователя.

Глава 9

Разработка приложения, напоминающего о задачах

В этой главе...

- Базовые требования к приложению
- Создание экранов приложения
- Создание деятельности со списком
- Идентификация намерения

Разработка приложений Android — увлекательное занятие, однако разработка мощных приложений Android с использованием множества сложных компонентов платформы — занятие еще более интересное. В этой главе вы начнете создавать приложение Task Reminder (Напоминание о задачах). Созданию данного приложения посвящены эта и несколько следующих глав.

Приложение Task Reminder позволяет пользователю создать список задач и ассоциировать с каждым элементом списка время, когда приложение должно напомнить пользователю о необходимости решения данной задачи.

Базовые требования к приложению

К приложению напоминания необходимо предъявить ряд базовых требований, чтобы оно делало то, что от него ожидается.

- ✓ Приложение должно позволять пользователю вводить текстовые данные. Должен же пользователь каким-то образом написать, что ему необходимо сделать, например, через месяц.
- ✓ Нужно сделать так, чтобы задачами было легко управлять.
- ✓ С каждой задачей должны быть ассоциированы дата и время напоминания.
- ✓ Пользователь должен быть извещен о том, что наступило время решения данной задачи.
- ✓ Пользователь должен иметь возможность удалить задачу.
- ✓ Приложение должно позволять пользователю не только добавлять, но и редактировать задачи.

Как видите, в данном приложении нужно запрограммировать много актов взаимодействия пользователя с операционной системой Android. В процессе создания приложения Task Reminder я познакомлю вас со многими аспектами среды разработки, которые помогут вам совершить головокружительную профессиональную карьеру.

Боевая тревога по расписанию

Чтобы в заданный момент времени прозвучал сигнал тревоги, нужно реализовать какую-то систему наблюдения за временем, аналогичную расписанию задач в Windows или команде `cron` в Linux. В операционной системе Windows пользователь может создать расписание задач, запускающих на выполнение программу или сценарий в заданные моменты времени. В Unix и Linux разработчик может использовать команду `cron` (сокращенно от древнегреческого "Кронос/Хронос" — время) для встраивания расписания задач в приложение.

Операционная система Android основана на ядре Linux 2.6, поэтому вполне естественно ожидать, что в ней есть редактируемый файл `crontab`, конфигурирующий команду `cron`. Но, к сожалению, в Android нет ни `cron`, ни `crontab`. Вместо них в Android используется класс `AlarmManager`, решающий аналогичную задачу. Класс `AlarmManager` позволяет задать время, когда приложение должно быть запущено операционной системой. Разработчик может задать один запуск или повторяющиеся запуски с определенным интервалом. В приложении Task Reminder класс `AlarmManager` используется для напоминания пользователям в определенные моменты времени о необходимости сделать что-либо.

Хранение данных

В процессе работы над приложением вы узнаете о многих новых для вас инструментах и средствах. Возможно, у вас уже сейчас возникло много вопросов относительно того, где мы разместим деятельности, данные задач, сигналы извещений и пр. Чтобы вы представили себе общую картину, сейчас вы узнаете, где размещаются компоненты приложения Task Reminder.

- ✓ **Деятельности и широковещательные приемники** — в одном пакете Java.
- ✓ **Данные задач** — в базе данных SQLite.
- ✓ **Информация о сигналах извещения** извлекается из базы данных SQLite и размещается в экземпляре класса `AlarmManager` с помощью системы намерений.

Деликатное напоминание

Когда сигнал извещения сработал, нужно каким-либо образом сообщить об этом пользователю. Платформа Android содержит средства переноса деятельности на передний план при генерации сигнала, но это не очень хороший способ извещения пользователя, потому что деятельность внезапно отберет фокус у активного средства, с которым работает пользователь. Например, если пользователь вводит номер телефона или разговаривает с абонентом, а в это время сигнал тревоги срабатывает и выводит на передний план другую деятельность, это приведет не только к тому, что пользователь будет застигнут врасплох, но и к тому, что у него будет прервано и, возможно, испорчено более важное дело. К тому же, пользователи не любят, когда происходит нечто такое, чего они не инициировали. Однако выводить деятельность на передний план и прерывать другие дела пользователя

не обязательно. Есть много других способов привлечь внимание пользователя, не отнимая фокус у текущей деятельности.

- ✓ **Уведомление** — это небольшое всплывающее окно, содержащее короткое сообщение для пользователя. Обычно уведомление присутствует на экране небольшое время, например несколько секунд. Уведомление никогда не получает фокус. В приложении Task Reminder уведомления используются не для напоминания о задаче, а для извещения о сохранении данных деятельностью. Пользователь чувствует себя комфортнее, когда видит, что устройство как-то отреагировало на его команду.
- ✓ **Класс `NotificationManager` (Менеджер извещений)** используется для извещения пользователя о том, что произошло некоторое событие или события. Извещение можно отобразить в строке состояния в верхней части экрана. Оно может состоять из нескольких представлений, визуально идентифицируемых с помощью созданных вами значков. Чтобы увидеть извещение, пользователь может переместить экран вниз.
- ✓ **Диалоговые окна.** Не очень популярный метод привлечения внимания пользователя состоит в открытии диалогового окна, которое может немедленно отобрать фокус у текущего выполняющегося приложения. Чаще всего пользователь будет недоволен тем, что приложение напоминания грубо прервало его работу с мобильным устройством. Если напоминаний много, они будут постоянно появляться на экране и мешать пользователю.

В приложении Task Reminder для обработки сигналов напоминаний используется класс `NotificationManager`.

Создание экранов приложения

Приложение Task Reminder имеет два разных экрана, на которых можно выполнять все базовые операции создания, чтения, обновления и удаления напоминаний о задачах. Первое представление отображает список всех текущих задач приложения поименно. Это же представление позволяет удалить задачу путем длинного нажатия на соответствующем элементе списка. Второе представление позволяет просматривать, добавлять и редактировать задачи. Каждый экран при необходимости взаимодействует с базой данных, в которой хранятся список и текущие параметры задач.

Создание нового проекта

Запустите программу Eclipse и создайте новый проект Android с целевой платформой Android 2.2 и минимальной версией SDK 4. Введите имена приложения, пакета и главной деятельности. Параметры создаваемого проекта приведены в табл. 9.1. Можете также открыть пример проекта Android для главы 9 в исходных кодах, приведенных на сайте данной книги. Тогда в Eclipse будет загружен проект с такими же параметрами, как в табл. 9.1.

Таблица 9.1. Свойства нового проекта

Свойство	Значение
Project Name (Имя проекта)	`Task Reminder`
Build Target (Целевая платформа)	Android 2.2 (API Level 8)
Application Name (Имя приложения)	`Task Reminder`
Package Name (Имя пакета)	`com.dummies.android.taskreminder`
Create Activity (Создать деятельность)	`ReminderListActivity`
Min SDK Version (Минимальная версия SDK)	`4`

Обратите внимание на имя деятельности `ReminderListActivity`. Обычно первой деятельности приложения присваивают имя `MainActivity`, однако в данном случае первый экран, который увидит пользователь, должен содержать список текущих задач. Фактически эта деятельность представляет собой экземпляр класса `ListActivity`, поэтому ей лучше присвоить имя `ReminderListActivity`.

Создание списка задач

При работе с классами `ListActivity` я предпочитаю, чтобы файл компоновки содержал слово `list` (список), тогда его легче найти при открытии папки `res/layout`. Переименуйте файл `main.xml`, расположенный в папке `res/layout`, присвоив ему новое имя `reminder_list.xml`. Для этого щелкните правой кнопкой мыши на имени файла на левой панели Eclipse и выберите в контекстном меню команду Refactor⇨Rename (Рефакторинг⇨Переименовать). Можете также выделить файл и нажать клавиши <Shift+Alt+R>.

Изменив имя файла компоновки, обновите его вызов `setContentView()` в файле `ReminderListActivity.java`. Для этого откройте файл `ReminderListActivity.java` и обновите ссылку на измененное имя файла.

Класс `ReminderListActivity` должен наследовать класс `ListActivity` вместо обычного базового класса деятельности `Activity`. Внесите и это изменение. Класс `ReminderListActivity` должен выглядеть, как в листинге 9.1.

Листинг 9.1. Класс `ReminderListActivity`

```
public class ReminderListActivity extends ListActivity {
  /** Вызывается при первом создании деятельности */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.reminder_list);
  }
}
```

Класс `ReminderListActivity` ссылается на ресурс компоновки `reminder_list`, который сейчас содержит код, сгенерированный по умолчанию при создании проекта. Чтобы компоновка взаимодействовала с классом `ListActivity`, нужно обновить файл компоновки, как показано в листинге 9.2.

Листинг 9.2. Содержимое файла `reminder_list.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
     "http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <ListView android:id="@+id/android:list"                    5
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>
      <TextView android:id="@+id/android:empty"               8
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/no_reminders"/>                    11
</LinearLayout>
```

Ниже приведено описание отмеченных строк кода.

- ✓ **5.** Определение экземпляра класса `ListView` — представления Android, используемого для отображения списка элементов, прокручиваемого по вертикали. Атрибут `id` элемента `ListView` должен иметь значение `@id/android:list` или `@+id/android:list`.
- ✓ **8.** Определение пустого состояния списка. Когда список пустой, должно отображаться представление, определенное в строке 8. Когда это представление отображается, элемент `ListView` автоматически скрывается, потому что в нем нет выводимых данных. Атрибут `id` этого представления должен иметь значение `@id/android:empty` или `@+id/android:empty`.
- ✓ **11.** В этом атрибуте строковый ресурс `no_reminders` (нет напоминаний) используется для извещения пользователя о том, что напоминаний (т.е. задач) в приложении в данный момент нет. Добавьте в файл `res/values/strings.xml` новый строковый ресурс с именем `no_reminders` и значением `Напоминаний нет`.

Создание и редактирование деятельностей задач

Приложению Task Reminder нужен еще один экран, позволяющий редактировать задачу. На этом экране должно быть все, что нужно для создания, чтения и редактирования задач в одной деятельности.

В рабочей среде Eclipse создайте новую деятельность, которая может выполнять все эти операции. Присвойте ей имя `ReminderEditActivity`. Для этого щелкните правой кнопкой мыши на имени пакета в папке `src` и выберите в контекстном меню команду New⇨Class (Создать⇨Класс). Можете также нажать клавиши <Shift+Alt+N> и выбрать команду Class. В окне нового класса в поле Superclass (Базовый класс) класса Java задайте класс **`android.app.Activity`** и щелкните на кнопке Finish (Готово).

В окне редактора Eclipse будет открыт пустой класс деятельности. Введите строки, отмеченные полужирным шрифтом.

```
public class ReminderEditActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.reminder_edit);
  }
}
```

В строке 5 этого класса компоновке деятельности присвоен ресурс `reminder_edit`, который будет определен в следующем разделе. Компоновка `reminder_edit` определяет несколько полей, с помощью которых пользователь может создавать или редактировать задачу.

Нужно известить платформу Android о существовании деятельности `ReminderEditActivity`, добавив ее в манифест. Для этого добавьте элемент `activity` в элемент `application` файла `AndroidManifest.xml`, как показано в следующем коде полужирным шрифтом.

```
<application android:icon="@drawable/icon"
           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>
```

Если не добавить деятельность в файл `AndroidManifest.xml`, то во время выполнения будет генерироваться сообщение об ошибке, информирующее о том, что операционная система Android не может найти класс данной деятельности.

Создание формы добавления и редактирования задач

Форма добавления и редактирования задач содержит следующие поля.

- ✓ **Название задачи.** Будет приведено в списке задач.
- ✓ **Текст задачи.** Более подробное описание того, что нужно сделать, получив напоминание.
- ✓ **Дата напоминания.** Дата, когда устройство должно напомнить пользователю о задаче.
- ✓ **Время напоминания.** Время, когда устройство должно напомнить пользователю о задаче.

На экране мобильного устройства или в эмуляторе форма добавления и редактирования задач выглядит, как показано на рис. 9.1.

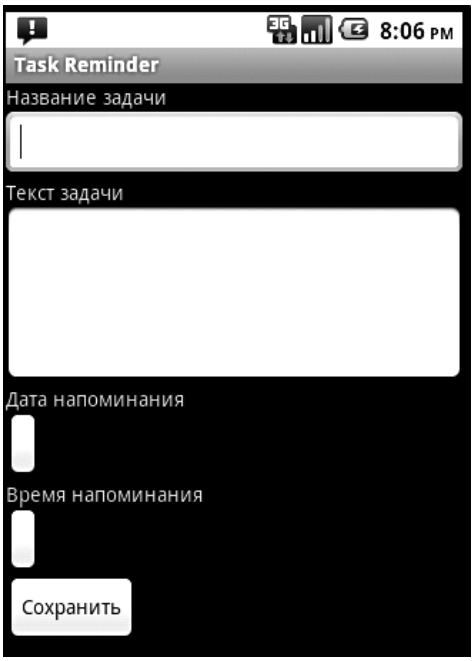


Рис. 9.1. Форма ввода параметров задачи

Чтобы отобразить эту форму на экране, создайте файл компоновки в папке `res/layout` и присвойте ему имя `reminder_edit.xml` (конечно, файлу можно присвоить любое имя, но будет лучше, если в вашем проекте имя файла будет совпадать с используемым в данной книге). Для этого выполните следующие операции.

1. **На левой панели Eclipse щелкните правой кнопкой мыши на папке `res/layout` и выберите в контекстном меню команду New⇨Android XML File (Создать⇨Файл Android XML).**
2. **В поле File введите имя файла `reminder_edit.xml`.**
3. **Переключатель Layout (Компоновка), определяющий тип ресурса, оставьте установленным.**
4. **Оставьте предложенное имя папки `res/layout`.**
5. **Выберите в раскрывающемся списке Select the root element (Выбор корневого элемента) пункт ScrollView (Прокручиваемое представление).**
6. **Щелкните на кнопке Finish (Готово).**

Теперь нужно создать все определения представлений, необходимых для отображения на экране элементов интерфейса, показанных на рис. 9.1. Для этого введите в файл `reminder_edit.xml` код, приведенный в листинге 9.3.

Листинг 9.3. Файл `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>
```

Ниже приведено короткое описание каждой строки кода, отмеченной номером.

- ✓ **5.** Определение родительского элемента `ScrollView`, который создает полосу прокрутки и позволяет прокручивать представление, когда его содержимое не помещается на экране. На рис. 9.1 экран показан в портретном режиме. Однако, если повернуть экран на 90°, представление тоже повернется, и больше половины представления будет отсечено. Родительский контейнер `ScrollView` позволяет при необходимости увидеть отсеченное содержимое с помощью прокрутки. Пользователь сможет, проведя пальцем вверх по экрану, переместить содержимое и увидеть отсеченную часть представления.
- ✓ **6.** Элемент `ScrollView` может содержать только один дочерний элемент. В данном случае в нем размещен контейнер `LinearLayout`, который содержит остальные элементы формы.
- ✓ **7.** Ориентация линейной компоновки определена как вертикальная, чтобы внутренние представления располагались одно под другим.
- ✓ **12.** Надпись для текстового поля с названием задачи. Чтобы компилятор не отметил эту строку как ошибочную, добавьте в файл `string.xml` строковый ресурс `title` со значением `Название задачи`.
- ✓ **15.** Текстовое поле, позволяющее ввести название задачи.
- ✓ **18.** Надпись для текстового поля с текстом задачи.
- ✓ **24.** Текстовое поле, позволяющее ввести текст задачи. Свойству `minLines` (минимальное количество строк) присвоено значение `5`, а свойству `gravity` (выравнивание) — значение `top` (вверх). Это означает, что поле всегда состоит как минимум из пяти строк и, когда пользователь начинает вводить текст, он размещается в верхней части представления.
- ✓ **27.** Надпись для текстового поля с датой напоминания. В надписи используется строковый ресурс. Поэтому добавьте в файл `strings.xml` строковый ресурс с именем `date` и значением `Дата напоминания`.
- ✓ **31.** Кнопка даты напоминания. При щелчке на ней активизируется календарь — элемент выбора даты `DatePickerDialog`, позволяющий задать дату с помощью встроенного диалогового окна Android. Когда дата выбрана, ее значение отображается на кнопке.
- ✓ **34.** Надпись для кнопки времени напоминания. В надписи используется строковый ресурс. Создайте его. Для этого добавьте в файл `strings.xml` строковый ресурс с именем `time` и значением `Время напоминания`.
- ✓ **38.** Кнопка времени напоминания. При щелчке на ней запускается элемент выбора времени `TimePicker`, позволяющий установить время с помощью встроенного окна Android. Когда время установлено, оно отображается на кнопке.
- ✓ **42.** Кнопка подтверждения, при щелчке на которой данные, введенные в форму, сохраняются.

Добавьте также в файл `strings.xml` **ресурсы** `body` и `confirm`, определяющие соответствующие надписи.

Создание деятельности со списком

Класс `ListActivity` отображает список элементов путем связывания с источником данных, например с массивом или таблицей базы данных, и запускает метод обратного вызова, когда пользователь выбирает элемент. Однако, чтобы создать список элементов, отображаемых на экране, нужно добавить компоновку, определяющую внешний вид каждой строки. Курсор базы данных (объект `Cursor`) предоставляет произвольный доступ чтения-записи к результирующему набору, возвращенному в ответ на запрос к базе данных.

Добавьте новый файл компоновки в папку `res/layout` с корневым элементом `TextView` и присвойте ему имя `reminder_row.xml`. Введите код, показанный в листинге 9.4.

Листинг 9.4. Файл `reminder_row.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
  xmlns:android=
     "http://schemas.android.com/apk/res/android"
  android:id="@+id/text1"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="10dip"/>
```

Данный код всего лишь определяет строку, в которой текстовое значение может размещаться с пустой полоской шириной 10 пикселей, независимо от разрешения экрана. В пятой строке определен идентификатор представления `text1`, на который мы будем ссылаться при загрузке списка с данными.

Представление `TextView` добавлено в Android как раз перед написанием данной книги. Если загрузить документацию Android и с помощью элементов управления хранилищем Android открыть разделы `simple_list_item_1` и `Android.R.layout`, посвященные спискам, то можно будет увидеть фактически это же определение XML. Источник можно найти по такому адресу:

```
http://android.git.kernel.org/?p=platform/frameworks/
base.git;a=blob;f=core/res/res/layout/simple_list_item_1.xml;
h=c9c77a5f9c113a9d331d5e11a6016aaa815ec771;hb=HEAD
```

Сокращенная версия доступна здесь:

```
http://bit.ly/9GzZzm
```

Для `ListActivity` нужно, чтобы содержимое представления списка заполнялось специальным адаптером. В принципе, доступно много адаптеров, но у нас в проекте на данный момент нет ни одного. В главе 12 мы создадим для приложения хранилище

на основе базы данных SQLite, а пока что воспользуемся фиктивными данными, чтобы можно было увидеть список в работе. Когда есть фиктивные данные, адаптер для `ListActivity` можно создать, вызвав метод `setListAdapter()`.

Создание фиктивных данных

В методе `onCreate()` класса `ReminderListActivity.java` после вызова `setContentView()` добавьте следующий код.

```
String[] items =
new String[] {"Задача1","Задача2","Задача3","Задача4"}; 1
ArrayAdapter<String> adapter =
   new ArrayAdapter<String>(this, R.layout.reminder_row,
          R.id.text1, items);                          4
setListAdapter(adapter);                               5
```

Ниже приведено краткое объяснение каждой инструкции, отмеченной номером строки.

- ✓ **1.** Создание массива строк, которые будут отображаться в списке.
- ✓ **4.** Создание адаптера `ArrayAdapter` строковых типов. Этот адаптер управляет представлением `ListView` с произвольным количеством любых объектов. В данном случае список содержит простой массив строк. В коде применяются обобщения Java, позволяющие разработчику задавать тип объекта, с которым работает адаптер `ArrayAdepter`. Конструктор `ArrayAdapter` принимает следующие параметры.
  - `this`. Текущий контекст (поскольку деятельность является реализацией класса `Context`, можно использовать экземпляр деятельности в качестве контекста).
  - `R.layout.reminder_row`. Компоновка строки, применяемая в `ListView` для каждого пункта списка.
  - `R.id.text1`. Идентификатор представления `TextView` в `R.layout.reminder_row`. В данном представлении размещаются значения массива.
  - `items`. Массив строк, загружаемый в `ListView`.
- ✓ **5.** Вызов `setListAdapter()`, сообщающий деятельности `ListActivity`, как нужно заполнить `ListView`. В данном случае для загрузки `ListView` используется адаптер `ArrayAdapter`, созданный в строке 4.

Запустите приложение Task Reminder на выполнение. Для этого выберите в Eclipse команду `Run`⇨`Run` (Выполнить⇨Выполнить) или нажмите клавиши <Ctrl+F11>. На экране эмулятора должен отображаться список `ListView` (рис. 9.2).

Предыдущий код иллюстрирует использование статического источника данных для отображения списка `ListActivity`. В главе 12 мы удалим этот код и загрузим содержимое `ListActivity` из базы данных SQLite.

Обработка событий щелчков

Элементы списка поддерживают события щелчков, что позволяет пользователю взаимодействовать с нужным пунктом списка. Базовый объект `View` определяет два главных типа событий щелчков.

Рис. 9.2. Список фиктивных задач

- ✓ **Короткий щелчок.** Пользователь короткое время прикасается к представлению на экране. Часто короткий щелчок называют просто "щелчок".
- ✓ **Длинный щелчок.** Пользователь прикасается к представлению на экране дольше нескольких секунд. Другое название — длинное нажатие.

Каждое представление и каждая деятельность могут перехватывать события щелчков посредством различных методов. В следующем разделе мы запрограммируем реакцию на каждый тип события в объекте `ListActivity`. В главе 11 я продемонстрирую вам реакцию на событие щелчка на кнопке `Button`.

Короткие щелчки

Базовый класс `ListActivity` операционной системы Android выполняет огромный объем работы по реагированию на события, чтобы уменьшить количество работы для программиста (т.е. для вас). Поэтому для реагирования на короткий щелчок нужно лишь определить метод обратного вызова. Введите следующий код в файл `ReminderListActivity.java` после метода `onCreate()`.

```
@Override
protected void onListItemClick(ListView l, View v,
                               int position, long id) {
  super.onListItemClick(l, v, position, id);
}
```

Этот код переопределяет заданную по умолчанию реализацию метода `onListItemClick()`, предоставляемую классом `ListActivity`. После щелчка на пункте списка этот метод вызывается и получает приведенные ниже параметры.

- ✓ `l`. Элемент `ListView`, на котором произошел щелчок.
- ✓ `v`. Пункт в `ListView`, на котором произошел щелчок.
- ✓ `position`. Позиция элемента списка, на котором щелкнул пользователь.
- ✓ `id`. Идентификатор строки в списке, на которой щелкнул пользователь.

С помощью этих переменных можно выяснить, на каком элементе списка произошел щелчок, и выполнить необходимые операции на основе этой информации. Когда пользователь щелкнул на элементе списка, должно быть инициировано намерение, которое активизирует деятельность `ReminderEditActivity`. Она позволяет редактировать задачу, ассоциированную с данным элементом списка.

Длинные щелчки

Длинный щелчок (иногда его называют *длинным нажатием*, или *длинным прикосновением*) происходит, когда пользователь прикасается к элементу интерфейса и не отпускает палец дольше нескольких секунд. Чтобы обработать событие длинного щелчка на элементе списка в деятельности `ListActivity`, добавьте следующий код в конец метода `onCreate()` в файле `ReminderListActivity`:

```
registerForContextMenu(getListView());
```

Внешний метод `registerForContextMenu()` регистрирует контекстное меню, выводимое для данного представления. Одно и то же контекстное меню может выводиться разными представлениями. Это означает, что каждый элемент списка может создавать и отображать контекстное меню. Метод `registerForContextmenu()` принимает в качестве параметра объект `View`, который должен быть зарегистрирован классом `ListActivity` как создающий контекстное меню. Внутренний метод `getListView()` возвращает объект `ListView`, используемый для регистрации. Метод `getListView()` является членом класса `ListActivity`.

Теперь, после регистрации класса `ListView`, создающего контекстное меню, нужно отреагировать на событие длинного щелчка на любом элементе списка. Когда в объекте `ListView` происходит длинный щелчок на некотором элементе списка, это событие распознается методом `registerForContextMenu()`, который вызывает метод `onCreateContextMenu()`. В данный момент контекстное меню готово к созданию и отображению. В методе `onCreateContextMenu()` оно конфигурируется.

В конце класса `ReminderListActivity` введите следующий метод.

```
@Override
public void onCreateContextMenu(ContextMenu menu,
                        View v, ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
}
```

В метод передаются следующие параметры.

- ✓ `menu`. Контекстное меню.
- ✓ `v`. Представление, для которого создается контекстное меню (т.е. на котором выполняется длинный щелчок).
- ✓ `menuInfo`. Дополнительная информация о пункте списка, для которого должно быть отображено контекстное меню. Состав информации может меняться в зависимости от типа представления в параметре `v`.

Внутри метода можете изменить меню, которое будет представлено пользователю. Например, когда пользователь совершает длинный щелчок на задаче, нужно, чтобы он имел возможность удалить ее. Следовательно, в контекстное меню нужно включить пункт Delete (Удалить). Мы добавим в меню пункт Delete в главе 10.

Идентификация намерения

Конечно, большинство реальных приложений гораздо сложнее, чем рассматриваемые в данной книге. Хотя многие приложения имеют всего два экрана (как Task Reminder), много чего происходит у них "за кулисами". Один из таких важных этапов взаимодействия между приложением и пользователем — активизация нового экрана в момент инициирования пользователем различных средств приложения. Как и в каждом мощном приложении, пользователь должен иметь возможность взаимодействовать с каждым экраном независимо от других. Следовательно, главная проблема состоит в том, как открыть другой экран.

Взаимодействие приложения с экранами обрабатывается системой намерений операционной системы Android. В главе 7 система намерений описана довольно подробно, однако в ней нет примера перехода с одного экрана на другой с помощью намерения. Рассмотрим эту процедуру сейчас.

Запуск новой деятельности с помощью намерения

Деятельности инициируются инфраструктурой намерений операционной системы Android. Класс `Intent` (Намерение) представляет сообщение, размещаемое в системе намерений Android (аналогично архитектурам на основе сообщений). Каждый объект, который может прореагировать на намерение, сообщает об этом платформе Android. В результате либо запускается деятельность, либо отображается список приложений, из которых пользователь выбирает нужное. Представляйте себе намерение как абстрактное описание операции, которую должно выполнить приложение.

Запустить конкретную деятельность несложно. В классе `ReminderListActivity` введите следующий код метода `onListItemClick()`.

```
@Override
protected void onListItemClick(ListView l, View v,
                               int position, long id) {
  super.onListItemClick(l, v, position, id);
  Intent i = new Intent(this,
                        ReminderEditActivity.class);       4
  i.putExtra("RowId", id);                                 5
  startActivity(i);                                        6
}
```

Ниже приведено краткое описание отмеченных строк кода.

- ✓ **4.** Эта строка создает новое намерение с помощью конструктора `Intent`, принимающего текущий контекст `this` и класс `ReminderEditActivity`, который система намерений должна попытаться запустить.
- ✓ **5.** Эта строка размещает дополнительные данные в объекте `Intent`. В данном случае в намерении размещается пара "ключ-значение". Ключом служит строка `RowId`, а значением — идентификатор представления, на котором щелкнул пользователь. Значение размещается в намерении таким образом, чтобы целевая деятельность

ReminderEditActivity могла извлечь его из объекта Intent и применить для загрузки информации о намерении. Сейчас мы используем фиктивные данные, поэтому после щелчка ничего не отображается, но в главе 12 мы передадим деятельности ReminderEditActivity реальные данные.

- ✓ **6.** Эта строка запускает новую деятельность из текущей. Инструкция вызова размещает сообщение намерения в системе намерений Android и предоставляет Android право решить, как должен быть открыт экран для пользователя.

Извлечение значений из предыдущих деятельностей

Иногда деятельность всего лишь запускается, и на этом роль операционной системы заканчивается. Никакие данные между деятельностями при этом не передаются. Но в некоторых случаях необходимо извлечь данные из входного намерения, чтобы решить, что нужно сделать. В приведенном выше коде мы ввели в намерение дополнительные данные RowId. В главе 12 значение RowId будет применено в объекте ReminderEditActivity для извлечения данных из базы данных SQLite и отображения их на экране.

Чтобы извлечь данные из входного намерения, введите следующий фрагмент кода в конец метода onCreate() целевой деятельности ReminderEditActivity.

```
if(getIntent() != null) {                                    1
  Bundle extras = getIntent().getExtras();                   2
  int rowId =
        extras != null ? extras.getInt("RowId") : -1;        3
  // Использование идентификатора строки  rowId
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Метод getIntent() предоставляется базовым классом Activity. Он извлекает входное намерение для деятельности. Необходимо убедиться в том, что метод не возвращает null, иначе он может вызвать крах приложения.
- ✓ **2.** Объект Bundle (пара ключ/значение) извлекается из намерения с помощью метода getExtras().
- ✓ **3.** Тернарный оператор используется для выяснения, получен ли объект Bundle. Если получен, то значение RowId извлекается из намерения, полученного от предыдущей деятельности с помощью вызова getInt(). Сейчас мы ничего не делаем со значением RowId, но в главе 12 мы используем его при создании запроса к базе данных SQLite для извлечения записи Task с целью ее редактирования.

Когда база данных SQLite будет установлена (это произойдет в главе 12), запись будет извлекаться из нее, а параметры задачи будут представлены пользователю на экране посредством формы редактирования, чтобы пользователь мог изменять задачу.

Создание окна выбора

На определенном этапе карьеры разработчика приложений Android у вас возникнет необходимость предоставить пользователю список приложений, которые могут обработать определенное намерение. Классический пример — обмен данными с друзьями с помощью общего сетевого инструмента, например электронной почты, SMS, Twitter, Facebook или Google Latitude.

В систему намерений Android встроены многие средства сетевого обмена. В приложении Task Reminder они не используются, но в будущем они могут быть вам очень полезными, поэтому я кратко упомяну о них. В листинге 9.5 приведен код, отображающий на экране окно выбора с доступными для пользователя вариантами.

Листинг 9.5. Создание окна выбора

```
Intent i = new Intent(Intent.ACTION_SEND);                    1
i.setType("text/plain");                                      2
i.putExtra(Intent.EXTRA_TEXT, "Привет, всем!");               3
i.putExtra(Intent.EXTRA_SUBJECT, "Моя тема");                 4
Intent chooser = Intent.createChooser(i,
                      "Кто должен сделать это?");             5
startActivity(chooser);                                       6
```

Ниже приведено объяснение каждой строки кода.

- ✓ **1.** Создание намерения, которое информирует Android о вашем желании передать что-либо по сети, например, по электронной почте.
- ✓ **2.** Тип содержимого сообщения. Можно задать любой тип MIME. Имена типов MIME чувствительны к регистру букв и всегда должны вводиться в нижнем регистре. Данная строка задает тип намерения. Следовательно, в окне выбора появятся только приложения, способные реагировать на данный тип намерения.
- ✓ **3.** Размещение дополнительных данных в намерении, в качестве которых служит тело сообщения, используемое приложением. Если будет выбрана клиентская программа электронной почты, эта строка станет телом электронного письма. При выборе Твиттера она станет сообщением. Каждое приложение, реагирующее на данное намерение, обрабатывает дополнительные данные по-своему. Не рассчитывайте на то, что данные будут обработаны определенным образом, потому что способ обработки полностью определяется типом целевого приложения. Разработчик текущего приложения должен позаботиться о том, чтобы данные были достаточно гибкими и не привели к краху целевое приложение.
- ✓ **4.** Задание специфичных данных аналогично строке 3, но на этот раз определяется тема сообщения. Если будет выбрана клиентская программа электронной почты, эти данные станут темой электронного письма.
- ✓ **5.** Создание окна выбора. Объект `Intent` имеет статический вспомогательный метод, создающий окно выбора. Выбор осуществляется объектом намерения. Вы предоставляете целевое намерение и заголовок окна выбора.
- ✓ **6.** Запуск намерения. С помощью этой инструкции создается и отображается на экране окно, позволяющее выбрать нужное приложение.

Окно выбора, созданное кодом листинга 9.5, показано на рис. 9.3.

Если система намерений не может найти ни одного приложения, способного обработать намерение, Android отображает окно сообщений, информирующее об этом (рис. 9.4).

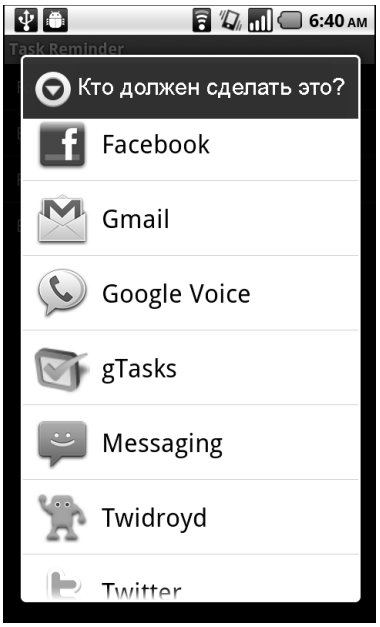


Рис. 9.3. Окно выбора; представлен список доступных программ обмена информацией

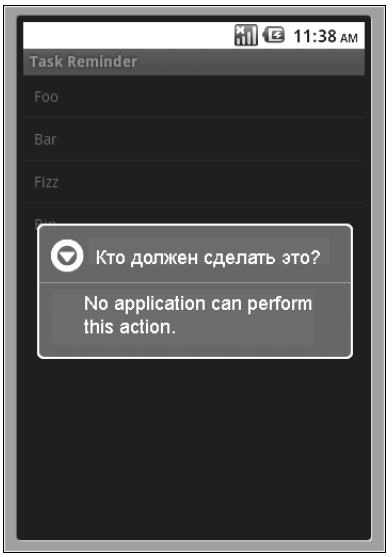


Рис. 9.4. Приложение информирует пользователя о том, что подходящие программы не найдены

Окна выбора — прекрасный инструмент взаимодействия приложений друг с другом. Если вызвать непосредственно метод `startActivity()` без создания окна выбора, приложение может потерпеть крах. Чтобы в листинге 9.5 запустить деятельность без окна выбора, нужно вместо `startActivity(chooser)` написать `startActivity(i)`. Однако данном случае приложение потерпит крах, потому что Android получает полную свободу дальнейших действий. Операционная система предполагает, что вы знаете, что делаете. Не создавая окно выбора, вы предполагаете, что на целевом устройстве есть как минимум одно приложение, способное обработать намерение. В противном случае Android сгенерирует исключение, видимое в DDMS и сообщающее о том, что ни один класс не может обработать намерение. Конечный пользователь, не имеющий среды разработки, видит этот процесс как крах приложения.

Чтобы сделать приложение более удобным и не подвергать пользователей стрессам, всегда создавайте окно выбора при запуске намерений, работающих с другими приложениями. Так принято делать в Android, и ваше приложение будет больше соответствовать тому, к чему привыкли пользователи.

Глава 10

Создание меню

В этой главе...

- Полезность меню
- Создание меню выбора
- Создание контекстного меню

Как жаль, что я сейчас нахожусь не в моем любимом мексиканском ресторане, просматривая меню с буррито и текилой! Вместо этого я в данный момент сам создаю меню, которое, однако, не имеет никакого отношения к вкусным блюдам. Я говорю о меню в приложении Android.

Операционная система Android предоставляет простой механизм добавления меню в приложение. Существуют следующие типы меню.

- ✓ **Меню выбора.** Используется чаще других типов меню, поскольку с его помощью чаще всего выбирается деятельность. Отображается на экране, в частности, когда пользователь нажимает на устройстве кнопку <MENU>. Могут быть отображены два типа меню выбора.
  - **Меню со значками.** Это меню выбора, отображаемое в нижней части экрана. Операционная система поддерживает до шести пунктов меню. С каждым пунктом ассоциированы значок и надпись. Флажки и переключатели в пунктах меню не поддерживаются.
  - **Расширенное меню.** Отображается, когда меню содержит более шести пунктов. Открывается при щелчке на значке More (Дополнительно), размещенном в последнем пункте меню со значками. Пункт More автоматически добавляется операционной системой в меню со значками, когда меню содержит более шести пунктов. Для дополнительных пунктов меню значки не создаются и не отображаются.
- ✓ **Контекстное меню.** Аналогично контекстному меню в приложениях Linux и Windows, но активизируется не правой кнопкой мыши, а длинным щелчком (пользователь держит палец на виджете более двух секунд).
- ✓ **Вложенное меню.** Список пунктов, открывающийся при выборе определенного пункта в меню выбора или контекстном меню. В Android 2.2 и ниже, в отличие от Linux и Windows, пункты вложенного меню не поддерживают вложенные меню следующих уровней.

В данной главе мы создадим меню выбора и контекстное меню. Если что-то не получается, загрузите полный код приложения Task Reminder с сайта книги.

Полезность меню

Если у вас есть устройство Android и вы уже загружали приложения с сайта Android Market, то вы, скорее всего, уже встречались как с хорошими, так и с плохими примерами реализации меню. Почему меню может быть плохим?

Плохим считается меню, которое предоставляет мало информации, вследствие чего им тяжело пользоваться. Ниже приведены наиболее распространенные недостатки меню:

- ✓ плохой заголовок, не отображающий назначение меню;
- ✓ меню без значков;
- ✓ меню, делающее не то, что в нем написано.

Из этих недостатков наиболее простительным, на первый взгляд, кажется отсутствие значков. Однако отнеситесь к этому серьезно. Отсутствие значков обычно означает, что разработчик не уделил достаточно времени пользовательскому интерфейсу и, следовательно, не позаботился о его эстетической привлекательности, полезности и легкости использования. Хорошее меню должно быть не только полезным, но и красивым. Внешний вид значков многое говорит об отношении разработчика к созданию приложения. Если значки информативные и красивые, значит, разработчик уделил много внимания не только внешнему виду меню, но и его работе. Учитывайте, что одного наличия значков недостаточно. Кроме того, они должны быть информативными и привлекательными.

Создание меню выбора

Меню можно создать либо в коде Java, либо в файле XML, расположенном в папке `res/menu`. Предпочтительный метод состоит в определении меню в файле XML и преобразовании его в программируемый объект, с которым можно взаимодействовать в коде. Этот метод помогает отделить определение меню от его фактического использования в коде приложения.

Создание файла XML

Чтобы определить меню с помощью файла XML, выполните следующие операции.

1. **Создайте в папке `res` вложенную папку `menu`.**
2. **Добавьте в нее файл `list_menu.xml`.**
3. **Введите в этот файл код, приведенный в листинге 10.1.**

Листинг 10.1. Меню для деятельности `ReminderListActivity`

```
<?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>
```

Обратите внимание на то, что в определении меню используется новый строковый ресурс. Вы создадите его в п. 4 данной инструкции. Значение `android:icon` ссылается на встроенный значок Android. Переопределять этот значок в ресурсе `drawable` не обязательно. В Android встроены все три версии значка в папках `ldpi`, `mdpi` и `hdpi`. Чтобы просмотреть другие доступные графические ресурсы, откройте документацию `android.R.drawable` по адресу `http://developer.android.com/reference/android/R.drawable.html`.

Все ресурсы класса `android.R` доступны для вас. Рекомендуется использовать их в разрабатываемых приложениях, чтобы пользовательские интерфейсы имели унифицированный внешний вид и приложения предоставляли пользователям привычную рабочую среду.

4. **В файле `strings.xml` создайте строковый ресурс с именем `menu_insert` и значением `Добавить задачу`.**
5. **Откройте класс `ReminderListActivity` и добавьте в файл следующий код.**

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);
  MenuInflater mi = getMenuInflater();
  mi.inflate(R.menu.list_menu, menu);
  return true;
}
```

В строке 4 метод получает от операционной системы объект `MenuInflater`, который нужен для создания объекта меню на основе файла XML. Эта операция выполняется методом `inflate` в следующей строке. Существующее меню — это объект меню, передаваемый в метод `onCreateOptionsMenu()`.

6. **Установите приложение в эмулятор и щелкните на кнопке MENU.**

На экране должно появиться меню с одним пунктом (рис. 10.1).

Обработка действий пользователя

Итак, меню создано. Теперь нужно запрограммировать какое-либо действие при выборе представленной в нем команды. Для этого введите следующий код в конец класса `ReminderListActivity`.

```
@Override
public boolean onMenuItemSelected(int featureId,
                                  MenuItem item) {      2
  switch(item.getItemId()) {                            3
    case R.id.menu_insert:                              4
      createReminder();                                 5
      return true;                                      6
  }
  return super.onMenuItemSelected(featureId, item);
}
```

Рис. 10.1. Меню со значком

Ниже приведено описание строк кода, отмеченных номерами.

- ✓ **2.** Метод, вызываемый при выборе пункта меню. Параметр `featureId` идентифицирует панель, на которой расположено меню. Параметр `item` идентифицирует пункт меню, на котором щелкнул пользователь.
- ✓ **3.** Идентификация пункта меню путем сравнения полученного идентификационного номера с известными пунктами меню. Инструкция `switch` используется для проверки каждого допустимого варианта. Идентификационный номер меню извлекается с помощью вызова метода `getItemId()` через объект `MenuItem`.
- ✓ **4.** Проверка идентификационного номера пункта меню на совпадение с изображенным на рис. 10.1.
- ✓ **5.** Если пользователь щелкнул на пункте Добавить задачу, приложение создает задачу путем вызова метода `createReminder()`, который мы определим в следующем разделе.
- ✓ **6.** Эта строка возвращает `true`, дабы проинформировать базовый метод `onMenuItemSelected()` о том, что выбор пункта меню обработан.

В данный момент строка `createReminder()` отмечается как ошибочная, но не беспокойтесь: это вызвано тем, что метод еще не создан. Займемся этим сейчас же.

Создание задачи

Метод `createReminder()` предназначен для перехода к деятельности `ReminderEditActivity`, с помощью которой пользователь создает новую задачу. Введите следующий код в нижнюю часть класса `ReminderListAcitvity`.

```
private static final int ACTIVITY_CREATE=0;
private void createReminder() {
  Intent i =
      new Intent(this, ReminderEditActivity.class);
  startActivityForResult(i, ACTIVITY_CREATE);
}
```

Этот код создает новое намерение, которое запускает деятельность `ReminderEditActivity`. Вызов `startActivityForResult()` применен потому, что после завершения деятельности нужны некоторые результаты. Приложение должно знать, когда деятельность завершается, чтобы предоставить коду возможность что-либо сделать. В данном случае по завершении деятельности `ReminderEditActivity` приложение должно повторно заполнить список задач. Инструкция вызова передает методу два параметра.

- ✓ **`Intent i`**. Намерение, запускающее деятельность `ReminderEditActivity`.
- ✓ **`ACTIVITY_CREATE`**. Код запроса, возвращаемый деятельности путем вызова `onActivityResult()`. Код запроса является константой уровня класса.

Приведенное ниже определение константы `ACTIVITY_CREATE` находится в классе `ReminderListActivity`.

```
private satatic final int ACTIVITY_CREATE=0;
```

Завершение деятельности

Последний вызов происходит после завершения деятельности `ReminderEditActivity`. Это вызов метода `onActivityResult()`. Ему передаются следующие параметры: код запроса, код результата и намерение, которое может содержать данные, передаваемые обратно вызывающей деятельности. Введите следующий код в конце класса `ReminderListActivity`.

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent intent)
{
  super.onActivityResult(requestCode, resultCode, intent);
  // Здесь нужно перезагрузить список
}
```

Сейчас этот код ничего не делает, но давайте оставим его без изменений до главы 12, в которой мы запрограммируем перезагрузку задач из базы данных SQLite. Ниже приведено описание передаваемых параметров.

- ✓ **requestCode.** Целочисленный код запроса, предоставленный в исходном вызове метода `startActivityForResult()`. Если текущая деятельность запускает множество дочерних деятельностей с разными кодами запроса, они (коды запроса) позволяют различать дочерние деятельности в инструкции `switch`. Это похоже на определение выбранных пунктов в методе `onMenuItemSelected()` с помощью инструкции `switch`.
- ✓ **resultCode.** Целочисленный код результата, возвращаемый дочерней деятельностью путем вызова метода `setResult()`. Код результата позволяет выяснить, было ли запрошенное действие завершено, отменено или прервано по какой-либо причине. Коды результата создаются программистом для выяснения, что произошло между вызовами деятельностей.
- ✓ **intent.** Намерение, которое дочерняя деятельность может создать для возврата результирующих данных вызывающей деятельности. Данные можно подключать к намерению с помощью метода `putExtra()`. В рассматриваемом примере используется тот же экземпляр намерения, который передается в метод `onActivityResult()`.

Базовый класс вызывается, чтобы не пропустить дополнительные операции, которые могут выполняться деятельностью.

Создание контекстного меню

При использовании устройства контекстное меню отображается, когда пользователь выполняет длинный щелчок на представлении. Контекстное меню появляется над текущей деятельностью и позволяет выбрать один из предлагаемых пунктов.

Контекстное меню создается почти так же, как и меню выбора. Его можно определить в файле XML и преобразовать в объект Java с помощью того же механизма `MenuInflater`. Для создания контекстного меню нужно, в первую очередь, вызвать метод `registerForContextMenu()` с целевым представлением. В главе 9 мы уже создали одно контекстное меню. После его создания нужно переопределить метод `onCreateContextMenu()`. Эту операцию мы тоже делали в главе 9.

В приложении Task Reminder необходим механизм удаления задачи, которая больше не нужна пользователю. Запрограммируем операцию удаления посредством контекстного меню. Пользователь выполняет длинный щелчок на задаче в списке задач, и операционная система Android отображает на экране контекстное меню, позволяющее удалить задачу путем щелчка на пункте меню.

Создание файла XML контекстного меню

Чтобы определить контекстное меню, нужно создать файл XML в папке `res/menu`. Присвойте файлу имя `list_menu_item_longpress.xml` и введите в него следующий код.

```
<?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>
```

Атрибут `title` содержит строковый ресурс `menu_delete`. Поэтому создайте в файле `strings.xml` строковый ресурс с именем `menu_delete` и значением `Удалить задачу`. Обратите внимание на то, что с данным меню не ассоциирован значок. Контекстные меню не поддерживают значки, потому что они всего лишь отображают списки доступных пунктов, выводимые над текущей деятельностью.

Загрузка меню

Чтобы загрузить контекстное меню, введите приведенный ниже код метода `onCreateContextMenu()` в класс `ReminderListActivity`.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater mi = getMenuInflater();
  mi.inflate(R.menu.list_menu_item_longpress, menu);
}
```

Этот код играет ту же роль, что и метод `onCreateOptionsMenu()` для меню выбора. Разница лишь в том, что он преобразует файл XML в объект контекстного меню, а не меню выбора. Теперь, если нажать пункт задачи на экране деятельности `ReminderListActivity` и подержать палец две секунды, на экране появится контекстное меню (рис. 10.2). Если контекстное меню не появляется, добавьте метод `onContextMenuItemSelected()`, как описано далее.

Обработка выбора пользователя

Реакция на выбор пользователем пункта контекстного меню программируется так же, как и для меню выбора. Чтобы приложение отреагировало на выбор пользователя, введите следующий код в конец класса `ReminderListActivity`.

```
@Override
public boolean onContextItemSelected(MenuItem item) {    2
  switch(item.getItemId()) {                             3
    case R.id.menu_delete:                               4
        // Удаление задачи
        return true;
  }
  return super.onContextItemSelected(item);
}
```

Рис. 10.2. Контекстное меню

Ниже приведено описание строк, отмеченных номерами.

- ✓ **2.** Заголовок метода, вызываемого при выборе пункта контекстного меню. Параметр `item` идентифицирует выбранный пункт.
- ✓ **3.** Инструкция `switch`, используемая для обработки выбранного пункта на основе идентификатора, определенного в файле `list_menu_item_longpress.xml`.
- ✓ **4.** Идентификационный код пункта `menu_delete` определен в файле `list_menu_item_longpress.xml`. Добавьте в файл `strings.xml` строковый ресурс `menu_delete` со значением `Удалить задачу`. Если выбран данный пункт меню, вставленный ниже (на месте строки 5) код выполняет операции, необходимые для удаления задачи. Пока что этот код ничего не делает. В главе 12 мы вставим в это место инструкции, удаляющие задачу из базы данных SQLite.

В файл `list_menu_item_longpress.xml` можно добавить много пунктов контекстного меню. Естественно, все они должны отображаться в инструкции `switch` метода `onContextmenuItemSelected()` и выполнять разные операции.

Глава 11

Обработка вводимых данных

В этой главе...

- Создание интерфейса ввода
- Выбор даты и времени
- Создание окна предупреждения
- Проверка вводимых данных

Приложения, в которых пользователь не вводит никаких данных, встречаются очень редко. Такими данными могут быть текст, выбор даты, выбор времени, установка переключателей или флажков и т.п. В каждом из этих случаев пользователь каким-либо образом взаимодействует с приложением. Вводить данные несложно, но, к сожалению, программирование ввода данных — гораздо более трудоемкая задача. В общем случае термин "ввод данных" означает очень широкий набор операций — щелчки на кнопках, ввод текста, перетаскивание, выбор пунктов меню, длинные щелчки и т.п., однако в данной главе рассматриваются только следующие операции: ввод текста, выбор даты или времени и реагирование на появление окон предупреждений.

Создание интерфейса ввода

Наиболее популярный инструмент ввода текстовых данных — виджет `EditText`. На других платформах он называется "текстовое поле". С помощью виджета `EditText` можно отобразить экранную клавиатуру или предоставить пользователю возможность вводить текст с помощью обычной клавиатуры.

Создание виджета `EditText`

В главе 9 мы создали компоновку представления в файле `reminder_edit.xml`, содержащую следующий код.

```
<EditText android:id="@+id/title"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content" />
```

Этот фрагмент кода определяет текстовое поле для ввода названия задачи. Текстовое поле отображается на экране, и пользователь можете вводить в него текст. По умолчанию виджет `EditText` занимает по ширине все доступное пространство на экране, а по высоте — столько пространства, сколько нужно для отображения содержимого. При щелчке на текстовом поле операционная система Android автоматически отображает экранную клавиатуру, с помощью которой пользователь может вводить текст. Ниже приведено более подробное определение виджета `EditText`, которое тоже есть в файле `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" />
```

В приложении Task Reminder приведенное выше текстовое поле применяется для ввода текста задачи. Ширина и высота виджета здесь такие же, как в предыдущем примере — по ширине виджет занимает все доступное пространство, а по высоте — столько, сколько нужно для отображения содержимого. Отличия от предыдущего текстового поля следующие.

- ✓ **`minLines`.** Это свойство определяет минимальную высоту представления `EditText`. Класс `EditText` является производным от класса `TextView` и наследует все его свойства, включая `minLines`. В приведенной выше разметке задана высота представления `EditText` не менее 5 строк. Благодаря этому представление выглядит как область, предназначенная для ввода не короткого названия, а длинного текста. Сравните с текстовыми полями в любой почтовой программе, где применяется тот же принцип: для короткой темы предоставлено однострочное текстовое поле, а для длинного тела письма — многострочное.
- ✓ **`scrollbars`.** Это свойство определяет, какие полосы прокрутки должны появляться, когда текст переполняет доступное пространство текстового поля. В данном случае задан вывод вертикальной полосы прокрутки.
- ✓ **`gravity`.** Когда пользователь передает фокус текстовому полю `EditText`, курсор ввода по вертикали располагается посередине представления. Однако пользователи интуитивно ожидают, что им будет предложено вводить текст с начала поля, т.е. курсор должен находиться вверху слева. Чтобы при выборе текстового поля курсор находился вверху, нужно присвоить атрибуту `gravity` значение `top`.

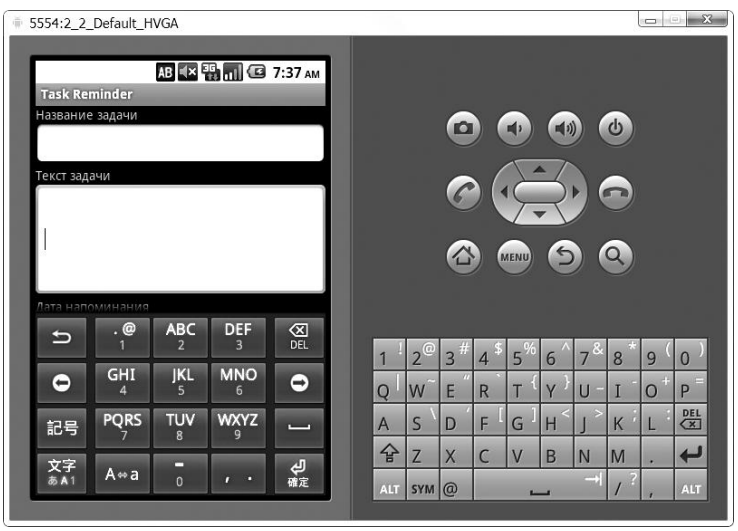


Рис. 11.1. Когда атрибут `gravity` не установлен, курсор находится посредине

Отображение экранной клавиатуры

Представление `EditText` очень гибкое, его можно сконфигурировать разными способами. В частности, оно определяет, как будет отображена экранная клавиатура. Во многих мобильных устройствах обычной клавиатуры нет, поэтому для ввода текста пользователю должен быть предоставлен другой механизм ввода. Атрибуты представления `EditText` позволяют программисту манипулировать визуальными свойствами экранной клавиатуры.

Разные текстовые поля могут задавать разные свойства экранной клавиатуры. Например, если в текстовое поле нужно вводить номер телефона, экранная клавиатура должна содержать только цифры. Буквы лишь зря займут драгоценное место и вынудят уменьшить клавиши цифр. Если текстовое поле предназначено для ввода электронного адреса, экранная клавиатура должна отображать символ `@`. Не забывайте, что вам предоставлено много способов конфигурирования экранной клавиатуры, с помощью которых можно повысить полезность и удобство приложения.

С помощью свойства `inputType` можно даже задать способ конфигурирования экранной клавиатуры. Свойств экранной клавиатуры так много, что для их описания потребуется целая книга. Полное описание этих свойств можно найти по такому адресу:

```
http://developer.android.com/reference/android/widget/
TextView.html#attr_android:inputType
```

Выбор даты и времени

Разрабатываемое нами приложение предназначено для напоминания пользователю в определенные моменты времени о необходимости решать некоторые задачи. Следовательно, приложение должно предоставлять возможность устанавливать дату и время, когда устройство напомнит пользователю о том, что нужно что-то сделать.

Если вы программировали задание даты и времени на других языках и платформах, у вас, скорее всего, остались неприятные воспоминания об этом процессе в связи с его запутанностью и трудоемкостью. Рад сообщить вам, что в Android добавлены инструменты, существенно облегчающие создание средств выбора пользователем даты и времени. Платформа Android предоставляет для этого два класса: `DatePicker` и `TimePicker`. Но и это еще не все. Каждый из этих классов предоставляет встроенные классы, которые отображают диалоговые окна выбора пользователем даты и времени. Вы можете либо встроить объект `DatePicker` или `TimePicker` в определенное представление, либо применить класс `Dialog`, избавив себя от необходимости создания представления, содержащего объект `DatePicker` или `TimePicker`.

Создание кнопок выбора даты и времени

Мы пока еще не добавили объекты `DatePicker` и `TimePicker` в приложение Task Reminder. Сделаем это сейчас. В файле компоновки `reminder_edit.xml` уже есть компоненты, готовые принять объекты `DatePicker` и `TimePicker`, — две кнопки с двумя надписями, определение которых приведено в листинге 11.1.

Листинг 11.1. Кнопки (пока что пустые) выбора даты и времени

```
<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"
  />
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Надпись `TextView` для кнопки выбора даты. Соответственно содержимому строкового ресурса `date`, отображается текст `Дата напоминания`.
- ✓ **4.** Определение кнопки, при щелчке на которой будет открыто диалоговое окно выбора даты `DatePickerDialog`, как показано в следующем разделе.
- ✓ **9.** Надпись для кнопки выбора даты. Отображен текст `Время напоминания`.
- ✓ **12.** Определение кнопки, при щелчке на которой будет открыто окно выбора времени `TimePickerDialog`.

Подключение класса выбора даты

Когда пользователь щелкает на кнопке, он должен получить возможность выбрать или отредактировать дату. Как это сделать, показано в следующем разделе.

Установка приемника щелчка на кнопке даты

Чтобы запрограммировать реакцию на щелчок на кнопке выбора даты, откройте файл деятельности, в которой должен быть размещен код обработки щелчка, — файл `ReminderEditActivity.java`.

В методе `onCreate()` введите следующий код:

```
registerButtonListenersAndSetDafaultText()
```

Программа Eclipse проинформирует вас о том, что нужно создать данный метод. Сделаем это сейчас. Легче всего сделать это так. Наведите указатель на строку, подчеркнутую красной волнистой линией, отмечающей ошибку, и выберите в появившемся меню команду создания метода `registerButtonListenersAndSetDefaultText()`. Введите в тело метода код, показанный в листинге 11.2.

Листинг 11.2. Реализация приемника щелчка на кнопке выбора даты

```
mDateButton.setOnClickListener(
                new View.OnClickListener() {                    1
  @Override
  public void onClick(View v) {                                 4
    showDialog(DATE_PICKER_DIALOG);                             5
  }
});
updateDateButtonText();                                         8
updateTimeButtonText();                                         9
```

Ниже приведено описание строк кода, отмеченных номерами.

- ✓ **1.** В этой строке используется переменная `mDateButton`. Мы еще нигде не определили ее, поэтому она отмечается как ошибочная. Ее нужно объявить в верхней части класса.

  ```
  private Button mDateButton;
  ```

  Вслед за объявлением переменной ее нужно инициализировать в методе `onCreate()` непосредственно после вызова метода `setContentView()`.

  ```
  mDateButton = (Button) findViewById(R.id.reminder_date);
  ```

  Далее можно устанавливать приемник `onClickListener()` для кнопки. Этот метод выполняется при щелчке на кнопке, т.е. в строке 5.
- ✓ **4.** Переопределение установленного по умолчанию поведения кнопки, чтобы в качестве реакции на щелчок можно было запрограммировать собственный набор действий.
- ✓ **5.** Эта строка определяет, что произойдет при щелчке на кнопке, а именно — будет вызван метод `showDialog()` через базовый класс деятельности. Метод принимает один параметр — идентификационный код окна, которое должно быть отображено. Идентификационный код — это значение, предоставляемое приложением в константе `DATE_PICKER_DIALOG`. Для даты и времени нужно определить две константы в верхней части класса, введя приведенный ниже код.

  ```
  private static final int DATE_PICKER_DIALOG = 0;
  private static final int TIME_PICKER_DIALOG = 1;
  ```

  Эти константы предоставляют идентификационные коды методу `showDialog()`, используемому для отображения окна выбора даты или времени.
- ✓ **8.** Вызов метода, обновляющего текст на кнопке даты. Метод будет создан далее в листинге 11.5.
- ✓ **9.** Вызов метода, обновляющего текст на кнопке времени. Этот метод тоже будет создан в листинге 11.5.

Реализация метода showDialog()

Метод `showDialog()` работает в базовом классе деятельности. При вызове метода `showDialog()` с некоторым идентификационным кодом автоматически вызывается метод `onCreateDialog()`, принадлежащий данной деятельности. Чтобы задать реакцию приложения на вызов метода `showDialog()`, введите в нижней части класса `ReminderEditActivity` код, приведенный в листинге 11.3.

Листинг 11.3. Реагирование на вызов `showDialog()` с помощью метода `onCreateDialog()`

```
@Override
protected Dialog onCreateDialog(int id) {                2
  switch(id) {
    case DATE_PICKER_DIALOG:                             4
      return showDatePicker();
  }
  return super.onCreateDialog(id);
}

private DatePickerDialog showDatePicker() {             10
  DatePickerDialog datePicker =
       new DatePickerDialog(ReminderEditActivity.this,
       new DatePickerDialog.OnDateSetListener() {       13
    @Override
    public void onDateSet(DatePicker view, int year,
                          int monthOfYear,
              int dayOfMonth) {                          17
    mCalendar.set(Calendar.YEAR, year);                  19
    mCalendar.set(Calendar.MONTH, monthOfYear);
    mCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);    21
    updateDateButtonText();                              22
    }
  }, mCalendar.get(Calendar.YEAR),
     mCalendar.get(Calendar.MONTH),
     mCalendar.get(Calendar.DAY_OF_MONTH));              25
  return datePicker;                                     26
}

private void updateDateButtonText() {                    29
  SimpleDateFormat dateFormat =
         new   SimpleDateFormat(DATE_FORMAT);            30
  String dateForButton =
          dateFormat.format(mCalendar.getTime());        31
  mDateButton.setText(dateForButton);                    32
}
```

Ниже приведено описание наиболее важных, отмеченных строк кода.

- ✓ **2.** Метод `onCreateDialog()` переопределяется и вызывается при вызове метода `showDialog()`. Методу `onCreateDialog()` передается параметр `id` — идентификационный код, который перед этим передавался методу `showDialog()`.
- ✓ **4.** Эта строка кода выясняет, совпадает ли идентификационный код, переданный в метод `onCreateDialog()`, с идентификационным кодом, переданным в `showDialog()`. Если код равен значению `DATE_PICKER_DIALOG`, данный метод возвращает значение, полученное от метода `showDatePicker()`.
- ✓ **10.** Определение метода `showDatePicker()`, возвращающего объект `DatePickerDialog`.
- ✓ **13.** В этой строке создается экземпляр класса `DatePickerDialog`, принимающий текущий контекст в качестве первого параметра. В качестве класса `Context` используется текущий экземпляр `ReminderEditActivity.this`. Приведено полностью квалифицированное имя, поскольку экземпляр находится во вложенном выражении. Следующий параметр — `onDateSetListener()` — задает функцию обратного вызова, определенную в строках 13–22. Эта функция возвращает выбранное значение даты. Другие параметры для `DatePickerDialog` перечислены в строке 25.
- ✓ **17.** Реализация метода `onDateSet()`, вызываемого, когда пользователь устанавливает дату с помощью `DatePickerDialog`, и щелкает на кнопке сохранения даты. Этот метод принимает следующие параметры.
  - **`DatePicker view`.** Инструмент выбора даты, используемый в диалоговом окне.
  - **`int year`.** Задаваемый год.
  - **`int monthOfYear`.** Задаваемый месяц. Месяцы нумеруются с 0 до 11 для совместимости с объектом `Calendar`.
  - **`int dayOfMonth`.** Задаваемый день месяца.
- ✓ **19-21.** В этом фрагменте кода используется переменная `mCalendar` типа `Calendar`. Она определена на уровне класса и позволяет отслеживать дату и время, выбираемые пользователем в деятельности `ReminderEditActivity` с помощью окон `DatePickerDialog` и `TimePickerDialog`. В строках 19–21 для изменения значений объекта `Calendar` используются методы `set`. Создайте определение переменной `mCalendar` в верхней части класса с помощью следующего кода.

  ```
  private Calendar mCalendar;
  mCalendar = Calendar.getInstance();
  ```

  Первую инструкцию поместите на уровне класса, а вторую — в тело метода `onCreate()`. Метод `getInstance()` возвращает экземпляр объекта `Calendar` и служит для инициализации объекта.
- ✓ **22.** После обновления объекта `mCalendar` нужно обновить текст на кнопке, щелкнув на которой пользователь открыл окно

DatePickerDialog. Обновление текста выполняется путем вызова метода updateDateButtonText(), определение которого приведено в строках 29–31.

- ✓ **25.** Здесь приведены остальные параметры конструктора окна DatePickerDialog. Приведенные значения отображаются в окне при его открытии. Для извлечения года, месяца и дня используется метод get объекта mCalendar. Если объект mCalendar не был установлен, он содержит значения текущей даты. Если объект mCalendar установлен и пользователь щелкает на кнопке для изменения даты, в окне DatePickerDialog отображается дата, хранящаяся в объекте mCalendar. Эта же дата считается установленной по умолчанию.
- ✓ **26.** В конце этого метода возвращается экземпляр класса Dialog, нужный методу onCreateDialog(). Класс DatePickerDialog наследует класс Dialog, поэтому можно вернуть объект DatePickerDialog. Это позволяет методу onCreateDialog() создать диалоговое окно, которое пользователь видит на экране.
- ✓ **29.** Как показано в строке 22, метод updatgeDateButtonText() вызывается после присвоения объекту mCalendar нового значения даты. Метод updateDateButtonText() обновляет текст на кнопке установки даты, на которой пользователь щелкает, когда хочет изменить дату. В этом методе тексту кнопки присваивается дата, выбранная пользователем, чтобы он, не открывая окно DatePickerDialog, видел, когда устройство напомнит о задаче.
- ✓ **30.** Установка объекта SimpleDateFormat, используемого для форматирования строки даты в конкретном классе. Дата чувствительна к региональным параметрам устройства и типу календаря (григорианский, юлианский, древнееврейский и др.). С помощью ключей форматирования даты, приведенных в документации Java (http://download-llnw.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html), дату можно отобразить в любом виде. В данном примере локальная константа DATE_FORMAT используется как параметр форматирования в объекте SimpleDateFormat. Эта константа определяет формат даты, видимый на экране устройства пользователем. Определите константу DATE_FORMAT в верхней части класса ReminderEditActivity следующим образом:

```
private static final String DATE_FORMAT = "yyyy-MM-dd";
```

- ✓ Формат "yyy-MM-dd" означает, что четыре цифры обозначают год, две цифры — месяц и еще две — день месяца. Значения отделяются дефисами. Например, дата 22 сентября 2010 года отображается как 2010–09–22.
- ✓ **31.** В этой строке объект SimpleDateFormat используется для форматирования даты mCalendar путем вызова метода getTime() через объект mCalendar. Метод возвращает строковый объект даты, преобразованный в формат DATE_FORMAT, установленный в объект SimpleDateFormat в строке 30.

✓ **32.** Установка текста на кнопке выбора даты с помощью метода `setText()` класса `Button`. Текстовому свойству кнопки присваивается значение, вычисленное в строке 31.

Теперь экземпляр `DatePickerDialog` подключен к приложению и может принимать данные, вводимые пользователем.

Подключение класса выбора времени

Класс `TimePickerDialog` предоставляет пользователю возможность выбрать время дня, когда устройство должно напомнить о задаче.

Установка приемника щелчка на кнопке

Подключение `TimePickerDialog` выполняется почти аналогично подключению `DatePickerDialog`. В первую очередь для кнопки нужно объявить приемник `onClickListener()`. Для этого объявите локальную переменную `mTimeButton` в верхней части класса `ReminderEditActivity` с помощью следующего кода:

```
private Button mTimeButton;
```

Инициализируйте переменную кнопки в теле метода `onCreate()` следующим образом:

```
mTimeButton = (Button) findViewById(R.id.reminder_time);
```

Теперь у нас есть кнопка редактирования времени, и для нее можно установить приемник щелчка. Введите код, приведенный в листинге 11.4, в тело метода `registerButtonListenersAndSetDefaultText()` после приемника щелчка на кнопке редактирования даты.

Листинг 11.4. Реализация приемника щелчка на кнопке редактирования времени

```
mTimeButton.setOnClickListener(
                  new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    showDialog(TIME_PICKER_DIALOG);
  }
});
```

Код метода такой же, как для кнопки даты, за исключением того, что при вызове метода `showDialog()` используется константа для времени, а не даты. Соответственно, при вызове метода `onCreateDialog()`, который выполняется при вызове метода `showDialog()`, ему будет передан идентификационный код времени, а не даты. В этот момент можно с помощью инструкции `switch` выбрать создание объекта `TimePickerDialog`. Кроме того, в верхней части класса должна присутствовать константа `TIME_PICKER_DIALOG` (сейчас она уже присутствует).

Теперь вернитесь к методу `onCreateDialog()` и добавьте следующий код после инструкции `return showDatePicker()`.

```
case TIME_PICKER_DIALOG:
  return showTimePicker();
```

Создание метода showTimePicker()

Полное определение метода приведено в листинге 11.5.

Листинг 11.5. Метод showTimePicker()

```
private TimePickerDialog showTimePicker() {
  TimePickerDialog timePicker =
              new TimePickerDialog(this,
          new TimePickerDialog.OnTimeSetListener() {     3
    @Override
    public void onTimeSet(TimePicker view,
                          int hourOfDay, int minute){    5
      mCalendar.set(Calendar.HOUR_OF_DAY, hourOfDay);    6
      mCalendar.set(Calendar.MINUTE, minute);            7
      updateTimeButtonText();                            8
    }
  }, mCalendar.get(Calendar.HOUR_OF_DAY),               10
      mCalendar.get(Calendar.MINUTE), true);            11
  return timePicker;
}
```

Код листинга 11.5 почти идентичен методу `showDatePicker()`, за исключением того, что на этот раз речь везде идет о времени, а не дате. Ниже приведено описание отмеченных строк.

- ✓ **3.** В класс `TimePickerDialog` добавляется новый метод `OnTimeSetListener()`, вызываемый, когда пользователь устанавливает время с помощью `TimePickerDialog`.
- ✓ **5.** Когда время установлено, часы и минуты передаются в `onTimeSet()`, что позволяет выполнять необходимые операции над значениями времени.
- ✓ **6.** Установка часа дня в объекте `Calendar`, объявленном на уровне класса.
- ✓ **7.** Установка минут часа в объекте `Calendar`, объявленном на уровне класса.
- ✓ **8.** Эта строка делегирует обновление текста кнопки установки времени методу `updateTimeButtonText()`, который определен далее в листинге 11.6.
- ✓ **10.** Эта строка определяет часы, применяемые по умолчанию в `TimePickerDialog`. Данное значение извлекается из объекта `Calendar`, существующего на уровне класса.
- ✓ **11.** Эта строка определяет минуты, применяемые по умолчанию в `TimePickerDialog`. Значение минут извлекается из объекта `Calendar`, существующего на уровне класса. Последнему параметру присвоено значение `true`, при котором время показывается в формате 24 часов (вместо 12 с указанием половины суток `p.m.` или `a.m.`).

В конце метода экземпляр `TimePickerDialog` возвращается методу `onCreateDialog()`, что позволяет вывести диалоговое окно на экран.

В строке 8 вызывается метод `updateTimeButtonText()`. Он аналогичен методу `updateDateButtonText()`, рассмотренному ранее. Различие между ними лишь в том, что метод `updateTimeButtonText()` обновляет текст на кнопке установки времени, а не даты. Код этого метода приведен в листинге 11.6. Введите его в нижней части класса `ReminderEditActivity`.

Листинг 11.6. Метод `updateTimeButtonText()`

```
private void updateTimeButtonText() {
  SimpleDateFormat timeFormat =
       new SimpleDateFormat(TIME_FORMAT);                2
  String timeForButton =
        timeFormat.format(mCalendar.getTime());          3
  mTimeButton.setText(timeForButton);                     4
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **2.** Создание объекта `SimpleDateFormat`, необходимого для форматирования значения времени. Формат времени задается строковой константой `TIME_FORMAT`, которую нужно объявить в верхней части класса `ReminderEditActivity` следующим образом:

  ```
  private static final String TIME_FORMAT = "kk:mm";
  ```

  Эта константа информирует класс `SimpleDateFormat` о том, что часы и минуты значения `Calendar` нужно разделить двоеточием, например `12:45 p.m.`
- ✓ **3.** Форматирование отображения времени, как задано в строке 2.
- ✓ **4.** Обновление текста на кнопке выбора времени. На кнопке отображается строка `timeForButton`, созданная в строке 3.

Теперь у нас установлены и сконфигурированы диалоговые окна выбора даты и времени, позволяющие пользователю задать их для редактируемой задачи. Пользователю не обязательно вручную вводить значения даты и времени. Он может выбрать их в диалоговом окне, отображающем календарь.

Создание окна предупреждения

Во многих случаях возникает необходимость проинформировать пользователя о том, что произошло нечто важное. Это можно сделать, отобразив диалоговое окно. Операционная система Android поддерживает диалоговые окна многих типов, позволяя запрограммировать в приложении любую необходимую реализацию окна.

Ниже описаны наиболее распространенные типы диалоговых окон.

- ✓ **Окно предупреждения.** Сообщает пользователю о том, что произошло нечто такое, на что он должен отреагировать. Позволяет установить текстовые значения на кнопках и определить действия, выполняемые

при щелчке на каждой кнопке. Как разработчик, вы можете отобразить в окне предупреждения список пунктов и позволить пользователю выбрать нужный пункт. Базовый класс окна предупреждения — `AlertDialog`.

- ✓ **Окно индикатора выполнения.** Используется для отображения кругового или полосного индикатора выполнения, сообщающего о проценте выполнения длительной фоновой задачи. Базовый класс окна индикатора выполнения — `ProgressDialog`.
- ✓ **Пользовательское окно.** Диалоговое окно, созданное и сконфигурированное программистом (т.е. вами). Базовый класс — `Dialog`. Можно создать в коде Java пользовательский класс, наследующий класс `Dialog`, или определить пользовательское окно в файле компоновки XML.

Зачем нужны диалоговые окна

Приходилось ли вам когда-либо работать с приложением, которое не сообщает ни о чем и не предупреждает о неблагоприятных событиях? Вполне возможно, что никогда не приходилось, потому что программисты всегда добавляют в приложения средства оповещения о важных событиях. Представьте себе почтовую программу, которая не извещает пользователя о поступлении нового электронного письма или о наличии непрочитанных писем. Пользователю придется постоянно думать: “Не забыл ли я заглянуть в почтовый ящик?” Оповещение пользователя о существенных событиях или о том, что он должен сделать некоторый выбор, — важная часть поведения мобильного устройства. Ниже приведен ряд ситуаций, в которых обычно используются диалоговые окна для извещения пользователя о новых сообщениях или о необходимости что-либо сделать.

- ✓ Выполняется фоновый процесс, в котором что-то происходит (например, загружается большой файл и объект `ProgressDialog` отображает процент загрузки).
- ✓ Значение, введенное в `EditText`, неправильное (например, номер телефона содержит буквы).
- ✓ Сетевое соединение разорвано, и сеть стала недоступной.
- ✓ Пользователь должен задать дату и время.
- ✓ Состояние телефона стало несовместимым с приложением. Например, для нормальной работы приложения должна быть включена система GPS или вставлена карта SD. Обычно подобные проблемы обнаруживаются автоматически при запуске приложения.
- ✓ Пользователь должен выбрать один из предложенных вариантов, перечисленных в списке, который отображается на экране.

Приведенный выше список возможных ситуаций далеко не полный, но он дает представление о том, что можно и нужно делать с диалоговыми окнами.

Работая с блокирующими фоновыми процессами любого типа (загрузка большого файла, длительная процедура решения какой-либо задачи и т.п.), вы всегда должны предоставлять пользователю диалоговое окно или индикатор выполнения, благодаря которым пользователь может видеть, что происходит в устройстве и скоро ли это кончится. В противном случае пользователь подумает, что приложение зависает, и не будет использовать его. Платформа Android предоставляет много типов индикаторов выполнения. Наиболее популярные — `ProgressDialog` и `ProgressBar`.

В данной книге в связи с ограниченностью ее объема не рассматривается полезный класс `AsyncTask`, который позволяет управлять длительными фоновыми задачами во время обновления пользовательского интерфейса. Хорошее руководство по этому классу можно найти по такому адресу:

`http://d.android.com/resources/articles/painless-threading.html`

Новый асинхронный поток можно также создать в коде вручную, однако использование класса `AsyncTask` существенно упрощает эту задачу.

Выбор диалогового окна для фоновой задачи

В каждом конкретном случае вы должны выбрать тип диалогового окна, оптимальный для данной задачи. Облегчить выбор оптимального диалогового окна поможет следующая инструкция.

1. **Долго ли выполняется данная задача?**
   - **Да.** Используйте класс `ProgressDialog`, создающий окно, которое сообщает пользователю о том, что выполняется некоторая фоновая задача, дабы он не подумал, что приложение зависло. Описание класса `ProgressDialog` можно найти в документации Android по адресу `http://d.android.com/guide/topics/ui/dialogs.html#ProgressDialog`.
   - **Нет.** Перейдите к п. 2.
2. **Нужно ли пользователю выполнять сложные операции в диалоговом окне?**
3. Под сложными операциями здесь понимаются задачи, которые нельзя решить с помощью простого диалогового окна `AlertDialog`.
   - **Да.** Создайте пользовательский класс диалогового окна, наследующий базовый класс `Dialog`. Можете также определить диалоговое окно с помощью файла компоновки XML. Дополнительную информацию о пользовательских диалоговых окнах можно найти в документации Android по адресу `http://d.android.com/guide/topics/ui/dialogs.html#CustomDialog`.
   - **Нет.** Перейдите к п. 3.
4. **Должен ли пользователь ответить, например, на такой вопрос: “Уверены ли вы в этом?” и щелкнуть на кнопке Да или Нет?**
   - **Да.** Создайте диалоговое окно `AlertDialog` и отреагируйте на щелчки на кнопках, переопределяя метод обратного вызова `onClickListener()`.
   - **Нет.** Перейдите к п. 4.

5. **Должен ли пользователь выбрать один вариант из списка доступных вариантов?**
   - **Да.** Создайте диалоговое окно предупреждения `AlertDialog`.
   - **Нет.** Перейдите к п. 5.
6. **Нужно ли всего лишь известить или предупредить пользователя о чем либо?**
   - **Да.** Создайте простое окно `AlertDialog`.
   - **Нет.** В данной ситуации диалоговое окно не нужно. Создайте сообщение для пользователя каким-либо другим способом.

Создание окна предупреждения

Иногда в приложении возникает необходимость сообщить пользователю нечто важное путем активизации диалогового окна. На платформе Android это легко сделать с помощью класса `AlertDialog.Builder`, который позволяет создать окно `AlertDialog` с разными параметрами и кнопками. Отреагировать на щелчки пользователя на кнопках можно в методах обратного вызова `onClickListener()` для каждой кнопки.

В приложении Task Reminder класс `AlertDialog.Builder` не используется, однако, ввиду его полезности во многих случаях, я продемонстрирую, как он используется в листинге 11.7.

Предположим, пользователь щелкнул на кнопке Сохранить в редакторе задач приложения Task Reminder. В ответ на щелчок можно запрограммировать вывод диалогового окна (рис. 11.2), которое просит подтвердить сохранение задачи.

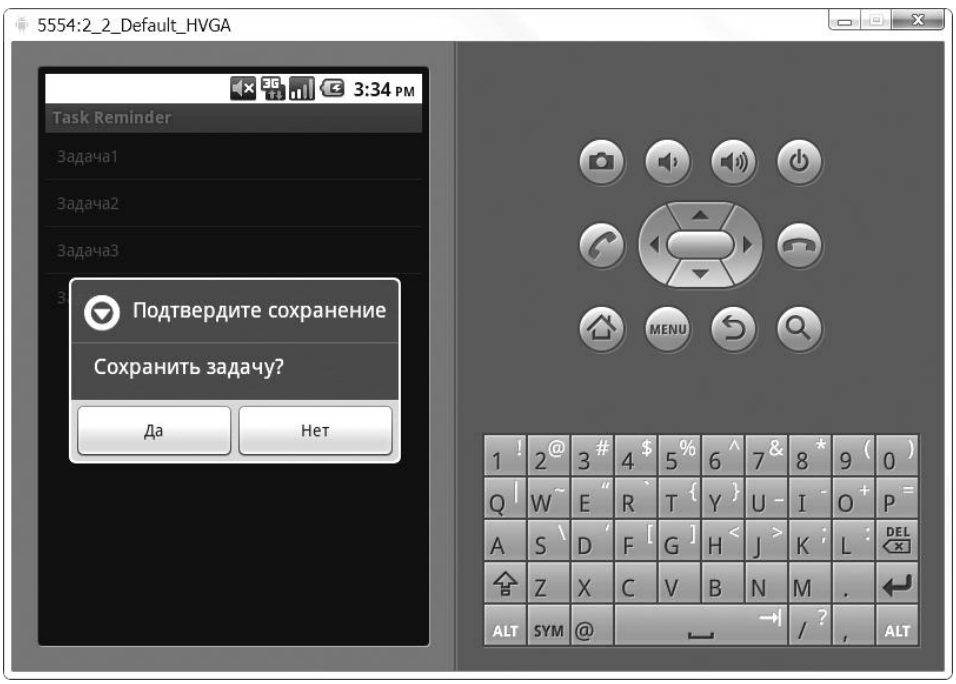


Рис. 11.2. Диалоговое окно предупреждения `AlertDialog`

Чтобы активизировать данное диалоговое окно, определите окно предупреждения, как показано в листинге 11.7. Фрагмент, приведенный в листинге 11.7, можно вставить в любое подходящее место кода, например в приемник щелчка на кнопке Сохранить.

В большинстве случаев рекомендуется отображать диалоговые окна с помощью методов `showDialog()` и `onCreateDialog()`, но в данном примере для краткости я создал диалоговое окно внутри приемника щелчка на кнопке Сохранить.

Листинг 11.7. Создание окна `AlertDialog` с помощью класса `AlertDialog.Builder`

```
AlertDialog.Builder builder =
  new AlertDialog.Builder(ReminderEditActivity.this);    2
builder.setMessage("Сохранить задачу?")                  3
   .setTitle("Подтвердите сохранение")                   4
   .setCancelable(false)                                 5
   .setPositiveButton("Да",                              6
      new DialogInterface.OnClickListener() {            7
       public void onClick(DialogInterface dialog,
                           int id) {
     // Здесь вставьте выполняемый код                   9
       }
       })
       .setNegativeButton("Нет",
             new DialogInterface.OnClickListener() {     12
          public void onClick(DialogInterface dialog,
                int id) {dialog.cancel();                14
       }
});
builder.create().show();                                 17
```

Можете вставить этот код в любое место приложения, в котором нужно отобразить диалоговое окно. Ниже приведено описание отмеченных строк кода.

- ✓ **2.** Создание экземпляра класса `AlertDialog.Builder` с контекстом `this`. В данном случае контекстом служит текущая деятельность.
- ✓ **3.** Задание сообщения, отображаемого в окне `AlertDialog` (см. рис. 11.2). Данное значение может быть строкой или строковым ресурсом.
- ✓ **4.** Задание заголовка окна `AlertDialog`. Как и предыдущее, это значение может быть строкой или строковым ресурсом.
- ✓ **5.** Атрибуту `Cancelable` присваивается значение `false`. Это означает, что пользователь обязательно должен сделать предлагаемый выбор и не может закрыть окно `AlertDialog`, вернувшись в прежнее состояние.
- ✓ **6.** Установка текста на кнопке подтверждения положительного ответа пользователя. На этой кнопке пользователь щелкает, когда хочет

выполнить операцию, указанную в сообщении диалогового окна. Соответственно, на кнопке написано Да. Этот текст можно задать как строку или строковый ресурс.

- ✓ **7.** Фрагмент кода между строками 7-11 представляет собой определение метода `onClickListener()` кнопки Да. Представленный в этих строках код выполняется после щелчка на кнопке Да. Комментарий в строке 9 указывает, куда нужно вставить выполняемый код.
- ✓ **12.** Установка надписи Нет на второй кнопке. Щелчок на кнопке означает, что пользователь не хочет выполнять операцию, указанную в окне `AlertDialog`. Текст надписи на кнопке можно задать либо как строку, либо как строковый ресурс.
- ✓ **14.** Заголовок приемника щелчка на кнопке Нет. Приемником служит метод `onClickListener()`. Метод ссылается на отображаемое в данный момент диалоговое окно. Для закрытия диалогового окна, когда пользователь щелкает на кнопке Нет, вызывается метод `cancel()` объекта `Dialog`.
- ✓ **17.** Эта инструкция создает объект диалогового окна с помощью метода `create()` и отображает его на экране путем вызова метода `show()`. В результате окно `AdertDialog` отображается со всеми заданными выше надписями.

Создать диалоговое окно `AlertDialog` с помощью класса `AlertDialog.Builder` значительно легче, чем пользовательское диалоговое окно с помощью класса `Dialog`. Конечно, окно `Dialog` может быть намного более мощным и гибким, чем `AlertDialog`. По этим причинами применяйте окно `Dialog` лишь в тех случаях, когда окно `AlertDialog` не предоставляет всех средств, необходимых в конкретной ситуации.

Когда пользователь щелкает на кнопке Сохранить (или любой другой кнопке, к которой вы подключите код, приведенный в листинге 11.7), на экране отображается окно `AlertDialog` с параметрами, установленными в листинге 11.7. В данный момент деятельность `ReminderEditActivity` временно неработоспособная, поэтому сохранить задачу пока что невозможно. Для эксперимента можете подключить этот код к какому-нибудь методу в деятельности `ReminderListAcitvity`. В главе 12 будет продемонстрировано сохранение задачи в базе данных SQLite.

Информацию о классе `Dialog` можно найти в документации Android по такому адресу:

```
http://d.android.com/guide/topics/ui/dialogs.html
```

Проверка вводимых данных

Предположим, вы создали форму, в которую пользователь может вводить информацию, и механизм сохранения введенной информации в базе данных или на удаленном сервере. Но что произойдет, если пользователь из-за невнимательности введет неправильные данные? Чтобы в базу данных попало как можно меньше ошибочной

информации, рекомендуется реализовать программную проверку правильности введенных данных.

Процедура проверки анализирует введенные данные перед их сохранением. Если пользователь забыл ввести название задачи и пытается сохранить ее, нужно ли позволить ему сделать это? Конечно, нет!

К сожалению, в операционной системе Android нет встроенных средств проверки вводимых данных. Возможно, хотя и маловероятно, что она появится в следующих версиях Android. А пока что нам придется вручную программировать инструкции проверки. Это можно делать многими способами.

Можно вставить в код инструкции, проверяющие введенные данные любым доступным способом. Ниже приведен ряд средств платформы Android, облегчающих проверку данных.

- ✓ **Класс `TextWatcher`.** Его можно реализовать в виджете `EditText` таким образом, что он будет выполнять обратный вызов при каждом изменении текста в окне виджета. Следовательно, вы сможете проверять текст при каждом нажатии клавиши ввода.
- ✓ **Событие `OnSave`.** Когда пользователь пытается сохранить форму, которую он заполняет, перехватите событие сохранения и проверьте программно все поля формы с помощью операторов сравнения. Если программа обнаружит ошибку, сообщите пользователю о результате проверки.
- ✓ **Метод `onFocusChanged()`.** Проверьте значения в полях формы, когда событие потери фокуса инициирует вызов метода `onFocusChanged()`. В этом методе весьма удобно размещать инструкции проверки.

В приложении Task Reminder нет средств проверки вводимых данных, однако их можно добавить любым описанным выше способом.

Уведомления

Довольно популярный способ оповещения пользователя о чем-либо, происходящем в системе, состоит в отображении всплывающего окна `Toast`, которое называется *уведомлением* и содержит заданный текст. Однако учитывайте, что уведомление отображается на экране всего несколько секунд, поэтому использовать его для сообщения об ошибке нежелательно.

Чтобы вывести на экран уведомление, содержащее, например, сообщение `Заполните поле титула`, нужно всего лишь вставить в нужное место кода следующий фрагмент.

```
Toast.makeText(ReminderEditActivity.this,
    "Заполните поле титула",
    Toast.LENGTH_SHORT).show();
```

По умолчанию уведомление отображается на экране всего несколько секунд. Если пользователь в это время отвлечется, он не увидит уведомление и не узнает, что уведомление когда-либо появлялось. Уведомление можно сконфигурировать таким образом, чтобы оно отображалось дольше, но тогда пользователю придется ждать, когда оно будет скрыто.

Другие способы проверки данных

Уведомления — не единственный способ оповещения пользователя о проблемах. Существует еще несколько популярных способов.

- ✓ **Окно `AlertDialog`.** Вставьте в код инструкции проверки и, если они обнаружили в данных ошибку, создайте и выведите на экран окно `AlertDialog`, сообщающее об этой ошибке. По сравнению с уведомлениями этот метод имеет преимущество: окно `AlertDialog` не исчезнет, пока пользователь не увидит его и не щелкнет на кнопке, расположенной в нем.
- ✓ **Подсветка поля ввода.** Добавьте в код проверку содержимого виджета `EditText` и, если содержимое неправильное, измените цвет фона виджета на красный. Увидев красный фон, пользователь поймет, что поле содержит ошибку. Можете также подать звуковой сигнал.
- ✓ **Пользовательские инструменты проверки.** Если вы планируете профессионально заняться разработкой приложений Android, создайте собственную библиотеку средств проверки. Со временем библиотека будет увеличиваться, и в ней будут средства проверки разного рода информации в разных ситуациях. Проверка состоит из двух этапов: анализ данных и оповещение пользователя. Можете разбить библиотеку на две части. Вторая часть может содержать средства подсветки неправильных данных и определения небольших представлений со стрелочками, указывающими на ошибку. Это похоже на средства индикации ошибки, применяемые Google в окне регистрации, когда посетитель входит в систему впервые.

Выше рассмотрены наиболее популярные способы отображения результатов проверки. Вы можете изобрести новые способы. Впрочем, способов отображения результатов проверки существует довольно много. Например, в главе 14 будет рассмотрена панель оповещения, с помощью которой можно сообщать пользователю об ошибке во введенных данных. Панель оповещения часто используется для сообщения о проблемах с фоновыми службами. Она не мешает работающим приложениям, поэтому ее удобно применять для деликатного напоминания пользователю о том, что нужно что-либо сделать в фоновом приложении.

Глава 12

Хранение данных

В этой главе...

- Где лучше хранить данные?
- Получение разрешения пользователя
- Создание базы данных SQLite
- Создание и редактирование задач с помощью SQLite

В настоящее время большинство приложений нуждается в сохранении данных разных типов для дальнейшего использования. Это касается и приложения Task Reminder. Может ли оно быть полезным, не сохраняя информацию о задачах, о которых нужно напомнить пользователю? К счастью, платформа Android и язык Java предоставляют много надежных инструментов сохранения данных.

Эта глава посвящена созданию и обновлению баз данных SQLite. Я буду подробно пояснять то, что делаю, однако теории баз данных в этой главе мы почти не коснемся.

Если вы не знакомы с реляционными базами данных и языком SQL, рекомендую почитать простейший учебник по этим темам.

В данной главе вам придется вводить много кода. Чтобы уменьшить объем ручной работы и количество ошибок, можете использовать готовые файлы с исходным кодом, доступные на сайте книги.

Где лучше хранить данные

Хранить данные можно во многих местах. Оптимальное место хранения данных зависит от ситуации в разрабатываемом приложении. Например, многие приложения взаимодействуют с музыкальными файлами. Пользователи хотят воспроизводить эти файлы с помощью разных музыкальных программ, поэтому желательно хранить их в месте, к которому имеют доступ все приложения. Во многих приложениях используются сохраняемые конфиденциальные данные, такие как зашифрованный пароль и регистрационное имя пользователя. Конфиденциальные данные лучше хранить в безопасном локальном хранилище. Операционная система Android предоставляет разные варианты хранения данных, подходящие для любых ситуаций.

Варианты хранения

Платформа Android предоставляет ряд мест постоянного хранения данных.

- ✓ **Общие предпочтения.** Представляют собой закрытые хранилища с ограниченным доступом, в которых записаны пары “ключ-значение”. Подробно рассматриваются в главе 15.

- ✓ **Внутреннее хранилище.** Находится в устройстве. По умолчанию файлы, записанные во внутреннее хранилище, принадлежат конкретному приложению и доступны только для него. Ни другие приложения, ни пользователь устройства не могут обращаться к ним. Когда пользователь деинсталлирует приложение, файлы, записанные во внутреннее хранилище, удаляются.
- ✓ **Локальный кеш.** Иногда в приложении нужно не хранить данные постоянно, а временно кешировать их. Для этого необходимо создать кеш во внутреннем хранилище приложения с помощью метода `getCacheDir()` класса `Activtity` или `Context`. Однако учитывайте, что при нехватке памяти для внутреннего хранилища Android может удалить файлы локального кеша для освобождения пространства. Поэтому не рекомендуется сохранять в локальном кеше более 1 Мбайта информации.
- ✓ **Внешнее хранилище.** Каждое устройство Android поддерживает общее внешнее хранилище, которое можно использовать для записи и чтения файлов. Внешним хранилищем может быть съемная карта SD или постоянная внутренняя память. Файлы, записанные во внешнее хранилище, являются открытыми. Это означает, что любой пользователь и любое приложение могут изменять их. Никакие средства безопасности к внешнему хранилищу не применяются. Пользователь может изменять файлы с помощью приложения, управляющего файловой системой (например, проводника), или путем монтирования физического носителя как внешнего хранилища. Перед работой с внешним хранилищем проверьте его текущее состояние с помощью объекта `Environment`. Вызвав метод `getExternalStorageState()`, можно убедиться в доступности внешнего хранилища в данный момент.

  В версии Android 2.2 добавлен новый набор методов для работы с внешними хранилищами. Наиболее важный метод — `getExternalFilesDir()` класса `Context`. При вызове ему передается строковый параметр, помогающий определить тип данных, которые нужно сохранять, например музыкальные файлы, фотографии, рингтоны и т.п. Дополнительную информацию о внешнем хранилище можно найти в документации Android по такому адресу:

  ```
  http://d.android.com/guide/topics/data/
  data-storage.html#filesExternal.html
  ```
- ✓ **База данных SQLite.** Операционная система Android поддерживает все средства SQLite, являющейся упрощенной СУБД на основе языка SQL (Structured Query Language — язык структурированных запросов). Базы данных SQLite доступны на многих платформах, включая Android, iPhone, Windows, Linux и Mac, и встроены во многие мобильные устройства. В Android можно создавать и конфигурировать таблицы, читать и записывать данные, выполнять запросы SQL и т.п. В этой главе мы реализуем базу данных SQLite для хранения параметров задач, о которых приложение Task Reminder должно напомнить пользователю.

- ✓ **Сетевое хранилище.** Любой удаленный источник данных, к которому мобильное устройство имеет доступ. Например, сайт Flickr предоставляет библиотеку функций, позволяющих хранить изображения на своих серверах. Ваше приложение тоже может хранить изображения на Flickr. Кроме того, само приложение может быть компонентом или клиентской частью какого-либо популярного сетевого инструмента, такого как Twitter, Facebook или Basecamp. Чаще всего приложения сообщаются с сетевыми хранилищами с помощью протокола HTTP, но не исключен любой другой протокол, поддерживаемый программным интерфейсом сетевого хранилища.

Выбор способа хранения

Доступность многих вариантов постоянного хранения данных порождает задачу выбора оптимального варианта. Для каждого приложения важно найти вариант, наилучший в конкретной ситуации. Во многих приложениях используются одновременно разные типы хранилищ.

Например, если приложение взаимодействует с библиотекой стороннего производителя (такого, как Twitter), то желательно хранить локальную копию всех данных с момента последнего обновления записей на сервере, потому что сетевое соединение медленное и не обеспечивает стопроцентную надежность. Применение локального хранилища позволяет приложению оставаться работоспособным (с определенными ограничениями) в период между сеансами связи или до следующего обновления данных. Хранить информацию можно в локальной копии базы данных SQLite. Когда пользователь инициирует обновление данных, новые данные будут сохранены в базе данных SQLite.

Если приложение извлекает и сохраняет данные только на сетевом носителе, рекомендуется модифицировать его с помощью базы данных SQLite таким образом, чтобы оно было полезным в перерывах между сеансами сетевого соединения. Такой режим работы приложения называется *автономным*. В реальности отключение соединения происходит довольно часто. Если при отсутствии сетевого соединения приложение неработоспособно, вы наверняка получите нелестные отзывы на сайте Android Market (а также предложения устранить этот дефект). Чтобы приложение могло хранить свои данные локально, нужно затратить какое-то время и приложить определенные усилия, однако они окупятся сторицей благодаря увеличению функциональности приложения.

Получение разрешения от пользователя

Предположим, ваш сосед хочет сохранить в вашем гараже декорации новогоднего праздника до следующего Нового года. Естественно, он должен получить на это ваше разрешение, поскольку вы являетесь владельцем гаража. Этот же принцип справедлив и для Android. Для сохранения данных в любом месте устройства необходимо получить разрешение его владельца, т.е. пользователя. Но сохранение — далеко не единственная операция, на выполнение которой нужно получить разрешение от пользователя.

Влияние разрешений на полезность приложения

Когда пользователь загружает и устанавливает приложение с сайта Android Market, операционная система извлекает из файла манифеста список разрешений, необходимых для нормальной работы приложения. При каждом обращении приложения к чувствительным компонентам, таким как внешнее хранилище, соединение с Интернетом, информация о телефоне и пр., операционная система извещает пользователя о том, что приложение хочет получить доступ к этим компонентам. Пользователь должен решить, хочет ли он предоставить приложению доступ к указанному компоненту.

Если приложение будет часто обращаться к пользователю за разрешениями, ему это надоест и он задумается, не установить ли другое приложение. Представьте себе, что было бы, если бы приложение Silent Mode Toggle (разработанное в предыдущих главах книги) часто запрашивало разрешения получить координаты GPS, установить соединение с Интернетом и получить информацию об оборудовании. Приложению Silent Mode Toggle эти разрешения не нужны. Пользователь заметит, что приложение запрашивает лишние разрешения, и подумает, что оно либо неисправное, либо зараженное. В любом случае пользователь откажется от этого приложения и поищет в Android Market другое приложение, выполняющее аналогичные функции.

Я опубликовал довольно много приложений и знаю по собственному опыту, что чем меньше разрешений запрашивает приложение, тем большему количеству пользователей оно понравится — и они установят его. Если приложению не нужно ни единого разрешения, тем лучше. Значит, создайте приложение без разрешений.

Установка требуемых разрешений в файле манифеста

Если приложение нуждается в каком-то разрешении, его нужно добавить в файл `AndroidManifest.xml` в проекте. Для работы с локальной базой данных SQLite никакие разрешения не нужны, поэтому для приложения Task Reminder необходимы только два разрешения, приведенные ниже (подробнее мы поговорим о них в главе 13):

- ✓ `android.permission.RECEIVE_BOOT_COMPLETED`;
- ✓ `android.permission.WAKE_LOCK`.

Разрешение `RECEIVE_BOOT_COMPLETED` позволяет приложению узнать, что мобильный телефон перезагружается. Разрешение `WAKE_LOCK` позволяет телефону находиться в режиме ожидания звонка, когда выполняется некоторая фоновая операция. Эти процедуры рассматриваются подробнее в главе 13 при обсуждении планировщика заданий `AlarmManager`.

Приведенные выше разрешения встречаются довольно редко, поэтому кратко остановимся на нескольких чаще используемых разрешениях. Для нормальной работы большинства приложений Android необходим доступ к Интернету. Некоторые приложения записывают и читают данные с карт SD. Если в приложении нужны эти разрешения, добавьте их в файл манифеста:

- ✓ `android.permission.INTERNET`;
- ✓ `android.permission.WRITE_EXTERNAL_STORAGE`.

Добавить разрешение в файл AndroidManifest.xml можно одним из следующих двух способов.

- ✓ С помощью редактора разрешений в AndroidManifest.xml. Чтобы открыть его, выберите команду Add⇨Uses Permission (Добавить⇨Используется разрешение). Выберите нужное разрешение в раскрывающемся списке.
- ✓ Путем редактирования файла AndroidManifest.xml вручную. Я считаю этот способ предпочтительным. Добавьте элемент uses-permission в элемент element. Запрос на разрешение выглядит в XML следующим образом:

```
<uses-permission android:name= "android.permission.WAKE_LOCK />
```

Добавьте разрешения WAKE_LOCK и RECEIVE_BOOT_COMPLETED в приложение Task Reminder. Увидеть полный список доступных разрешений можно в документации Android:

```
http://d.android.com/reference/android/Manifest.permission.html
```

Если не объявить разрешения, необходимые приложению, и пользователь установит его на своем устройстве, приложение не будет работать, как ожидается. Иногда будут генерироваться исключения времени выполнения, приводящие к краху приложения. Поэтому всегда проверяйте, все ли разрешения вы объявили в файле манифеста. Впрочем, заметить эту ошибку легко, потому что приложение не будет работать в эмуляторе.

Создание базы данных SQLite

Для приложения Task Reminder необходимо место, в котором можно хранить информацию о задачах. Лучшее место для информации такого рода — локальная база данных SQLite. Приложение должно читать, создавать, обновлять и удалять задачи, о которых оно будет напоминать пользователю.

Как работает база данных SQLite

Две деятельности, определенные в приложении Task Reminder, выполняют ряд операций над базой данных SQLite. Деятельность ReminderEditActivity должна выполнять следующие операции.

1. Создание новой записи в базе данных.
2. Чтение записи с целью отображения данных на экране для редактирования.
3. Обновление существующей записи.

Деятельность ReminderListActivity выполняет следующие операции.

1. Чтение всех задач для отображения их на экране.
2. Удаление задачи в ответ на событие щелчка в контекстном меню, открывшемся в результате длинного щелчка на задаче.

Для работы с базой данных SQLite с ней нужно установить связь с помощью классов пакета `android.database`. Обычно стараются как можно сильнее абстрагировать взаимодействие с SQLite от объектов `Activity`. Механизмы базы данных рекомендуется размещать в другом файле Java (и даже в другом пакете, если компонент базы данных большой). Это необходимо для разбиения приложения на функциональные слои. Тогда, если необходимо изменить код, влияющий на базу данных, изменение коснется только одного места, а другие уровни абстракции не будут затронуты. Данный подход более подробно рассматривается в следующих разделах.

Создание файла Java с кодом базы данных

Создадим в проекте Android файл Java и поместим в него код, относящийся к базе данных. Присвоим этому файлу имя `RemindersDbAdapter.java`.

Адаптером принято называть класс оболочки, позволяющий несовместимым классам сообщаться друг с другом. Представляйте себе адаптер как переходное устройство между двумя несовместимыми устройствами, например между DVD-плеером и телевизором. Создав адаптер для обслуживания подключения к базе данных, вы общаетесь с ним посредством языка Java. Адаптер преобразует запросы на языке Java в команды SQLite и наоборот — результирующие таблицы SQLite в данные на языке Java.

Определение ключевых элементов

Перед открытием и созданием базы данных необходимо определить несколько ключевых полей. Создайте класс `ReminderDbAdapter` и введите в него код, показанный в листинге 12.1.

Листинг 12.1. Константы, поля и конструкторы класса `ReminderDbAdapter`

```
private static final String DATABASE_NAME = "data";      1
private static final String DATABASE_TABLE =
                                        "reminders";     2
private static final int DATABASE_VERSION = 1;           3

public static final String KEY_TITLE = "title";          5
public static final String KEY_BODY = "body";
public static final String KEY_DATE_TIME =
                                    "reminder_date_time";
public static final String KEY_ROWID = "_id";            8


private DatabaseHelper mDbHelper;                       11
private SQLiteDatabase mDb;                             12
private static final String DATABASE_CREATE =           14
  "create table " + DATABASE_TABLE + " ("
    + KEY_ROWID + " integer primary key autoincrement, "
    + KEY_TITLE + " text not null, "
    + KEY_BODY + " text not null, "
    + KEY_DATE_TIME + " text not null);";

private final Context mCtx;                             21
public RemindersDbAdapter(Context ctx) {                23
    this.mCtx = ctx;
  }
```

Ниже приведено краткое описание отмеченных строк кода.

- ✓ **1.** Имя базы данных, которое будет фигурировать в файловой системе Android.
- ✓ **2.** Имя таблицы, принадлежащей базе данных и содержащей информацию о задачах. Таблица будет подробно рассмотрена в следующем разделе.
- ✓ **3.** Версия базы данных. При обновлении схемы базы данных необходимо увеличивать этот параметр на единицу. Для этого должна быть создана реализация метода `onUpgrade()` класса `DatabaseHelper` (см. далее).
- ✓ **5-8.** Определение имен столбцов таблицы.
- ✓ **11.** Объявление переменной типа `DatabaseHelper` на уровне класса. Класс `DatabaseHelper` является реализацией класса `SQLiteOpenHepler` операционной системы Android.
- ✓ **12.** Объявленный на уровне класса экземпляр объекта базы данных SQLite, позволяющий создавать, читать, обновлять и удалять записи.
- ✓ **14.** Текст сценария, создающего базу данных. Определенные выше константы конкатенируются для создания столбцов. Каждый компонент сценария подробно описан далее.
- ✓ **21.** Объявление объекта контекста, который будет ассоциирован с объектом базы данных SQLite.
- ✓ **23.** Установка объекта контекста с помощью конструктора класса адаптера.

Теперь базу данных можно создать с помощью сценария `DATABASE_CREATE`, определенного в предыдущем коде.

Визуализация таблицы SQLite

Объект таблицы SQLite содержит данные, которыми должно управлять приложение Task Reminder. На экране мобильного устройства таблица не отображается, однако ее нетрудно представить себе или отобразить на экране компьютера в виде обычной таблицы со строкой заголовков. Каждая строка таблицы содержит данные, относящиеся к одной задаче, а каждый столбец — данные одного типа, принадлежащие разным задачам (рис. 12.1). Имена столбцов определены в строках 5–8 листинга 12.1. Реально данные хранятся в виде нулей и единиц, например 0100100101010010. Естественно, не имеет смысла просматривать такие данные визуально: человек ничего не видит в строке из нулей и единиц, поэтому очень полезным может быть представление в виде таблицы, показанное на рис. 12.1.

_id	title	body	reminder_date_time
1	Заказ билетов на сам	Посетить агентство, менеджер - Сн	15.05.2011 16:00
2	Расписание занятий	Составить расписание с участием	18.05.2011 12:00
3	Книга на верстке	Проследить за версткой книги	08.07.2011 15:00
4	Депозит в ERDE	Переоформить депозит в банке ER	16.08.2011 14:00

Рис. 12.1. Визуальное представление таблицы SQLite

В строке 14 листинга 12.1 начинается код, определяющий сценарий создания базы данных. Сценарий содержит ряд констант, определенных в файле класса специально для создания сценария. При запуске сценария в SQLite система управления базой данных создает таблицу `reminders` в базе данных `data`. Ниже приведено описание директив, создающих таблицу и столбцы.

- ✓ **`create table DATABASE_TABLE.`** Часть сценария, приказывающая SQLite создать таблицу базы данных с именем `reminders`.
- ✓ **`ROW_ID.`** Идентификатор задачи. Этому столбцу присвоены атрибуты `integer primary key autoincrement`. Атрибут `integer` означает, что в столбце хранятся целочисленные значения. Атрибут `primary key` говорит о том, что столбец `ROW_ID` служит в качестве первичного ключа таблицы. Атрибут `autoincrement` информирует SQLite о необходимости автоматически увеличивать значение на единицу каждый раз при вставке новой записи (т.е. новой задачи). Например, если существуют значения 1, 2 и 3, то при вставке в таблицу следующей записи в столбец `ROW_ID` будет вставлено значение 4.
- ✓ **`KEY_TITLE.`** Название задачи. Атрибут `text` сообщает SQLite о том, что в столбце хранятся текстовые строки. Атрибут `not null` означает, что столбец не может содержать значение `null`, т.е. в нем обязательно должно быть записано какое-либо значение.
- ✓ **`KEY_BODY.`** Описание задачи. Этому столбцу присвоены те же атрибуты, что и столбцу `KEY_TITLE`.
- ✓ **`KEY_DATE_TIME.`** В этом столбце хранятся даты и время напоминания. Атрибуты те же, что и у предыдущих двух столбцов. Возможно, вам это покажется странным, ведь в предыдущих двух столбцах хранятся текстовые строки, а не даты и время. Объясняется это тем, что в SQLite, в отличие от Java, нет данных типа даты и времени, поэтому приходится хранить их в полях текстового типа. Впрочем, это не порождает никаких проблем, потому что в коде Java несложно преобразовать любую отформатированную строку в соответствующий тип.

Информацию о работе с датами и временем в SQLite можно найти в документации баз данных по адресу `http://www.sqlute.org/datatype3.html#datetime`.

Создание таблицы

Теперь все готово для создания таблицы. Сделаем это с помощью реализации базового класса `SQLiteOpenHelper`. Введите в классе `RemindersDbAdapter` код, приведенный в листинге 12.2. Этот код создает вложенный класс Java внутри класса `RemindersDbAdepter`.

Листинг 12.2. Класс, создающий таблицу

```
  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
    // В данном приложении не используется. При
    // необходимости можете обновлять базу данных
    // с помощью директивы ALTER.
  }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Заголовок определения класса, реализующего базовый класс `SQLiteOpenHelper`.
- ✓ **3.** Вызов конструктора базового класса `SQLiteOpenHelper`. Конструктор конфигурирует создание и открытие базы данных и управление ею. Операции создания и открытия базы данных выполняются в момент вызова метода `getReadableDatabase()` или `getWriteableDatabase()` через экземпляр класса `SQLiteOpenHelper` (в данном случае — через переменную `mDbHelper`).
- ✓ **7.** Заголовок метода `onCreate()`, который вызывается при создании базы данных.
- ✓ **8.** Запуск сценария, который создает базу данных и таблицу. Метод `execSQL()` принимает в качестве строкового параметра сценарий SQL. Система управления базой данных SQLite выполняет данный сценарий в результате вызова метода `execSQL()`.
- ✓ **12.** Метод `onUpgrade()` можно использовать, когда нужно обновлять структуру существующей базы данных. В данном примере эта операция не рассматривается.

База данных создается путем вызова метода `getReadableDatabase()` или `getWritableDatabase()` через объект `DatabaseHelper`. Для этого введите следующий код в любое место класса `RemindersDbAdapter`.

```
public RemindersDbAdapter open()
            throws android.database.SQLException {
  mDbHelper = new DatabaseHelper(mCtx);
  mDb = mDbHelper.getWritableDatabase();
  return this;
}
```

Метод `open()` открывает (и при необходимости предварительно создает) базу данных с помощью только что созданного вами класса `DatabaseHelper`. Этот класс возвращает вызывающему классу себя же с помощью ключевого слова `this`. Это делается потому, что вызывающему классу (`ReminderEditActivity` или `ReminderListActivity`) необходим доступ к данным посредством класса `DatabaseHepler`. Фактически метод `open()` возвращает экземпляр класса `RemindersDbAdapter()`.

Закрытие базы данных

База данных — дорогостоящий ресурс, поэтому ее нужно закрывать, когда она не используется, тем более что в мобильных устройствах объем всех ресурсов довольно ограниченный. Чтобы закрыть базу данных, введите следующий метод в любое синтаксически допустимое место класса `RemindersDbAdapter`.

```
public void close() {
  mDbHelper.close();
}
```

При вызове этого метода приложение закрывает базу данных. Метод будет вызываться в классе `ReminderEditActivity`, когда пользователь отменяет деятельность с помощью кнопки Back (Назад) мобильного устройства.

Обновление базы данных

Когда необходимо обновлять базу данных? Рассмотрим следующую ситуацию. Вы поставили приложение на рынок, и 10 000 пользователей установили его и активно работают с ним. Предположим, оно им нравится. Многие пользователи даже передают запросы с предложениями добавить новые средства. Вы решили реализовать одно из предложений, но для этого нужно изменить схему базы данных. Это делается с помощью инструкции `ALTER` языка SQL в теле метода `onUpgrade()`, который обновляет базу данных, не затрагивая существующие данные. Можно также обновить базу данных путем удаления существующей и создания новой. Однако чаще всего этот способ нежелателен, потому что при удалении базы данных исчезают все данные пользователя. Представьте себе, что при обновлении приложения Task Reminder вы удалили все задачи, созданные пользователями. Естественно, они будут очень недовольны.

Создание и редактирование задач с помощью SQLite

В первую очередь нужно создать задачу. Для этого необходимо вставить запись в таблицу базы данных. После этого необходимо добавить задачу в список задач в классе `ReminderListActivity`, который позволяет редактировать задачу путем прикосновения к ее названию или удалить ее путем длительного прикосновения. В предыдущих главах мы уже рассмотрели указанные операции создания, чтения, обновления и удаления задач со стороны пользовательского интерфейса, теперь рассмотрим их с другой стороны — со стороны базы данных.

Вставка записи о задаче

Когда готовы все необходимые компоненты, вставить в таблицу запись о задаче несложно. Эта операция состоит из следующих этапов.

1. Объявление необходимых локальных переменных.
2. Создание приемника щелчка на кнопке Сохранить.
3. Извлечение значения из текстовых полей `EditText`.
4. Взаимодействие с классом `RemindersDbAdapter`.
5. Открытие и закрытие базы данных.

На данный момент вы уже знакомы с классом `SQLiteDatabase`, поэтому можем приступить к реализации класса `RemidnersDbAdapter`, выполняющего перечисленные выше операции.

Сохранение в базе данных значений, видимых на экране

Когда пользователь создает задачу, он размещает ее в классе `ReminderEditActivity`. Для создания задачи нужно объявить переменную на уровне класса `RemindersDbAdapter`, инициализируемую методом `onCreate()`. После инициализации мы откроем базу данных, вызвав метод `open()` класса `RemindersDbAdapter` в методе `onResume()`. Введите приведенный ниже код в класс `RemindersEditActivity`.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  mDbHelper = new RemindersDbAdapter(this);
  setContentView(R.layout.reminder_edit);
  // ... продолжение метода onCreate()
```

В данный момент у нас есть ссылка на класс `RemindersDbAdapter`, позволяющая вызвать его для создания задачи. Чтобы добавить задачу, нужно иметь ее название и описание, а также дату и время напоминания. Для получения доступа к названию и описанию добавьте три переменные на уровне класса `ReminderEditActivity`. Две из них имеют тип `EditText` и ссылаются на значение `EditText` в компоновке деятельности `ReminderEditActivity`. Еще одна переменная представляет кнопку Сохранить, при щелчке на которой задача сохраняется в базе данных. Добавим эти три переменные в верхнюю часть файла `ReminderEditActivity`.

```
private EditText mTitleText;
private Button mConfirmButton;
private EditText mBodyText;
```

Создадим экземпляры этих переменных в методе `onCreate()`.

```
mConfirmButton = (Button) findViewById(R.id.confirm);
mTitleText = (EditText) findViewById(R.id.title);
mBodyText = (EditText) findViewById(R.id.body);
```

Сейчас в деятельности есть объект `Calendar`, заполненный представлениями `DatePicker` и `TimePicker`. Следовательно, создавать какие-либо переменные или объекты для даты и времени не нужно. Осталось лишь обеспечить возможность сохранения задачи после ввода значений в текстовые поля `EditText` (название и описание). Сохранение выполняется в результате щелчка на кнопке `confirm`. Поэтому

нужно подключить к данной кнопке приемник щелчка, введя приведенный ниже код в метод регистрации приемника `registerButtonListenersAndSetDafaultText()`.

```
mConfirmButton.setOnClickListener(
                  new View.OnClickListener() {
  public void onClick(View view) {
    saveState();                                          3
    setResult(RESULT_OK);                                 4
    Toast.makeText(ReminderEditActivity.this,             5
    getString(R.string.task_saved_message),
      Toast.LENGTH_SHORT).show();
    finish();                                             7
  }
});
```

Ниже дано описание отмеченных строк кода.

- ✓ **3.** Вызов метода `saveState()`, который сохраняет задачу.
- ✓ **4.** Установка результата деятельности `ReminderEditActivity`. Как вы помните, деятельность `ReminderEditActivity` началась с вызова метода `startActivityForResult()`. Присвоение константы `RESULT_OK` возвращаемому значению информирует деятельность `ReminderListActivity` о том, что при вызове метода `finish()` все происходит, как запланировано. Константа `RESULT_OK` является членом базового класса `Activity`. Результирующий код можно просмотреть в методе `onActivityResult()` класса `ReminderListActivity`. Приложение может вернуть вызывающему методу любое количество результатов, чтобы метод мог принять решение, что делать дальше.
- ✓ **5.** Создание уведомления, извещающего пользователя о том, что задача сохранена. Для уведомления необходимо создать строковый ресурс с именем `task_saved_message` и значением Задача сохранена.
- ✓ **7.** Вызов метода `finish()`, который завершает деятельность `ReminderEditActivity`.

В классе `ReminderEditActivity` нужно создать метод `onSave()`, код которого приведен в листинге 12.3. Этот метод сообщается с классом `RemindersDbAdapter` для сохранения задачи.

Листинг 12.3. Метод `saveState()`

```
private void saveState() {
  String title = mTitleText.getText().toString();         2
  String body = mBodyText.getText().toString();           3

  SimpleDateFormat dateTimeFormat = new
    SimpleDateFormat(DATE_TIME_FORMAT);                   5
  String reminderDateTime =
    dateTimeFormat.format(mCalendar.getTime());           6
  long id = mDbHelper.createReminder(title, body,
                    reminderDateTime);                    8
}
```

Ниже приведено объяснение отмеченных строк кода.

- ✓ **2-3.** Извлечение текста из представлений `EditText`.
- ✓ **5.** Определение объекта формата даты `SimpleDateFormat`, который будет использоваться для записи даты и времени в базе данных SQLite. Формат хранится в строковой константе `DATE_TIME_FORMAT`. Ее нужно создать в верхней части класса, введя следующий код:

  ```
  public static final String DATE_TIME_FORMAT =
                              "yyyy-MM-dd kk:mm:ss";
  ```

  Эта строка задает формат, отображающий дату и время, например следующим образом: `2011-11-20 12:34:21`. В базе данных SQLite нет специального типа для даты и времени, поэтому они обычно хранятся в текстовом виде.
- ✓ **6.** Получение даты и времени и размещение их в локальной переменной.
- ✓ **8.** Создание напоминания о задаче с помощью метода `createReminder()` на уровне класса `ReminderDbAdapter`, представленного переменной `mDbHelper`. Мы создадим этот метод в классе `RemindersDbAdapter` в следующем разделе.

Задача создается путем извлечения значений текстовых полей `EditText` и локального объекта `Calendar` с последующим вызовом метода `createReminder()` через класс `RemindersDbAdapter`. Адаптер создает оболочку для логики базы данных SQLite, поэтому деятельности `ReminderEditActivity` ничего не нужно знать о том, как сохраняются и читаются данные в таблице базы данных.

Полная реализация класса `RemindersDbAdapter`

Вы покупали когда-нибудь автомобиль, только взглянув на несколько фотографий дверной ручки, сиденья, бампера и приборной панели? Конечно, нет! Покупая автомобиль, вы обязательно рассмотрите его сначала в целом, а потом — подробно каждый узел. Всегда лучше сначала увидеть целое, а потом внимательно рассмотреть детали. Это же справедливо и для реализации базы данных SQLite в классе `RemindersDbAdapter`.

Попытка объяснить принцип работы класса `RemindersDbAdapter` по частям наверняка потерпит неудачу. Поэтому в листинге 12.4 я приведу сначала полную реализацию класса `ReminderDbAdapter`, чтобы вы почувствовали, с чем мы будем работать. Затем я подробно остановлюсь на каждом новом фрагменте. О ссылках на класс `ReminderDbAdapter` мы еще поговорим далее, после чего сможем не возвращаться к этой теме в остальных главах книги.

Листинг 12.4. Код класса `ReminderDbAdapter`

```
public class RemindersDbAdapter {

  private static final String DATABASE_NAME = "data";
  private static final String DATABASE_TABLE =
                                      "reminders";
  private static final int DATABASE_VERSION = 1;
```

```
public static final String KEY_TITLE = "title";
public static final String KEY_BODY = "body";
public static final String KEY_DATE_TIME =
                                  "reminder_date_time";
public static final String KEY_ROWID = "_id";

private DatabaseHelper mDbHelper;
private SQLiteDatabase mDb;

private static final String DATABASE_CREATE =
  "create table " + DATABASE_TABLE + " (" + KEY_ROWID
    + " integer primary key autoincrement, "
    + KEY_TITLE + " text not null, "
    + KEY_BODY + " text not null, "
    + KEY_DATE_TIME + " text not null);";

private final Context mCtx;

public RemindersDbAdapter(Context ctx) {
  this.mCtx = ctx;
}

public RemindersDbAdapter open() throws SQLException {
  mDbHelper = new DatabaseHelper(mCtx);
  mDb = mDbHelper.getWritableDatabase();
    return this;
}

public void close() {
  mDbHelper.close();
}

public long createReminder(String title,
                           String body, String
    reminderDateTime) {                                  38
  ContentValues initialValues = new ContentValues();
  initialValues.put(KEY_TITLE, title);
  initialValues.put(KEY_BODY, body);
  initialValues.put(KEY_DATE_TIME, reminderDateTime);

  return mDb.insert(DATABASE_TABLE, null,
                                  initialValues);        44
}

public boolean deleteReminder(long rowId) {              47
  return mDb.delete(DATABASE_TABLE, KEY_ROWID +
                       "=" + rowId, null) > 0;           48
}

public Cursor fetchAllReminders() {                      51
  return mDb.query(DATABASE_TABLE,
```

```
                new String[] {KEY_ROWID,
                KEY_TITLE, KEY_BODY, KEY_DATE_TIME},
                null, null, null, null, null);
    }

    public Cursor fetchReminder(long rowId)
                          throws SQLException {          55
      Cursor mCursor =
        mDb.query(true, DATABASE_TABLE,
                  new String[] {KEY_ROWID, KEY_TITLE,
                  KEY_BODY, KEY_DATE_TIME}, KEY_ROWID +
                  "=" + rowId, null,null, null, null,
                  null);                                 56
      if (mCursor != null) {
        mCursor.moveToFirst();                           57
      }
      return mCursor;

    }

    public boolean updateReminder(long rowId,
         String title, String body, String
          reminderDateTime) {                            63
      ContentValues args = new ContentValues();          64
      args.put(KEY_TITLE, title);
      args.put(KEY_BODY, body);
      args.put(KEY_DATE_TIME, reminderDateTime);

    return mDb.update(DATABASE_TABLE, args, KEY_ROWID +
                "=" + rowId, null) > 0;                  69
    }
// Класс SQLiteOpenHelper для краткости опущен.
// Его код находится здесь.
}
```

Ниже приведено описание отмеченных новых строк кода.

- ✓ **38.** Заголовок метода `createReminder()`. Непосредственно под заголовком объект `ContentValues` используется для определения значений полей, из которых состоит вставляемая строка.
- ✓ **44.** Вставка строки в таблицу путем вызова метода `insert()`. Метод возвращает значение типа `long` — уникальный идентификатор вставленной строки. В классе `ReminderEditActivity` идентификатор присваивается локальной переменной, которая в главе 13 будет использована в классе `AlarmManager` для определения нужной задачи. Использование метода `insert()` и передаваемые в него параметры рассматриваются в следующем разделе.
- ✓ **47.** Заголовок определения метода `deleteReminder()`, который принимает один параметр `rowId` — идентификатор строки, которую нужно удалить.

- ✓ **48.** Применение значения `rowId` в вызове метода `delete()`, удаляющего запись из таблицы базы данных. Использование параметров метода `delete()` рассматривается далее.
- ✓ **51.** Заголовок определения метода `fetchAllReminders()`, который находит в базе данных и возвращает все задачи с помощью метода `query()`. Объект `Cursor` используется вызывающим приложением для извлечения значений из результирующего набора запроса, возвращенного методом `query()`. Использование метода `query()` и его параметров подробнее рассматривается далее.
- ✓ **55.** Заголовок определения метода `fetchReminder()`, который принимает один параметр — идентификатор задачи в таблице базы данных.
- ✓ **56.** Метод `query()` используется для возвращения объекта `Cursor`.
- ✓ **57.** Объект `Cursor` может содержать много строк, причем исходная позиция курсора может быть не на первой записи. Метод `moveToFirst()`, вызванный через объект `Cursor`, перемещает курсор на первую запись результирующего набора. Этот метод вызывается, только когда переменная курсора не равна `null`. Курсор не помещается немедленно в первую запись по той причине, что исходный результирующий набор не упорядочен. Чтобы работать с некоторой записью, нужно сначала перейти к ней. Представляйте себе результирующий набор как большую коробку, заполненную детскими игрушками. Чтобы что-нибудь делать с игрушкой, например рассмотреть ее со всех сторон, нужно вынуть ее из коробки.
- ✓ **63.** Заголовок определения метода `updateReminder()`, в котором используется стандартный метод обновления базы данных `update()`. Метод `update()` заполняет существующую запись задачи новой информацией. Его использование подробно рассматривается далее.
- ✓ **64.** Создание объекта `ContentValues`. В нем хранятся значения, которые нужно обновить в базе данных SQLite.
- ✓ **69.** Обновление записи базы данных с помощью новых значений, предоставленных пользователем приложения.

В предыдущем коде продемонстрировано использование методов создания, чтения, обновления и удаления записей базы данных SQLite. Каждый метод принимает разные параметры, которые рассматриваются далее.

Операция вставки записи

Это довольно простая операция, поскольку нужно всего лишь добавить новую строку в таблицу базы данных (термины *строка* и *запись* — синонимы) с помощью метода `insert()`, который принимает следующие параметры.

- ✓ **`table`.** Имя таблицы, в которую нужно вставить запись. В рассматриваемом примере в качестве имени таблицы используется константа `DATABASE_TABLE`.

- ✓ **nullColumnHack.** Синтаксис SQL не разрешает вставку совершенно пустой строки. Если следующий параметр (`values`) пустой, полям присваивается значение `NULL`. Для этого методу передается значение `null`, которое подставляется в пустое поле.
- ✓ **values.** В этом параметре передаются значения, вставляемые в запись. Методу передается локальная переменная `initialValues`, которая содержит пары "ключ-значение".

Операция чтения

Это наиболее часто выполняемая операция над таблицами баз данных. Для чтения используется стандартный метод `query()`, который возвращает результирующий набор на основе списка критериев, задаваемых вызывающим методом. Метод `query()` возвращает объект `Cursor`, предоставляющий неупорядоченный доступ чтения и записи к результирующему набору, возвращенному запросом к базе данных. Метод `query()` принимает следующие параметры.

- ✓ **distinct.** Каждая строка должна быть уникальной. Копии задач нам не нужны. Для этого параметру `distinct` присвоено значение `true`.
- ✓ **table.** Имя таблицы базы данных, к которой направлен запрос. Значение имени таблицы находится в константе `DATABASE_TABLE`.
- ✓ **columns.** Список столбцов, возвращаемых запросом. При значении `null` возвращаются все столбцы. Обычно так не делают, потому что передача лишних столбцов отрицательно влияет на быстродействие базы данных. В данном примере передается строковый массив, содержащий имена возвращаемых столбцов.
- ✓ **selection.** Фильтр возвращаемых строк, определяющий, какие строки не нужно возвращать. Фильтр отформатирован как директива `WHERE` без ключевого слова `WHERE`. Если передать `null`, будут возвращены все строки таблицы. В данном примере передается или `null`, или идентификатор задачи, определяющий, какую строку нужно извлечь из таблицы.
- ✓ **selectionArgs.** В фильтр возвращаемых строк можно включить знаки вопроса (`?`). Они будут заменены значениями строкового типа, приведенными в `selectionArgs`. В рассматриваемом примере параметры `selectionArgs` не нужны, поэтому методу передается значение `null`.
- ✓ **groupBy.** Фильтр группирования строк, отформатированный как директива `GROUP BY` без ключевых слов `GROUP BY`. При передаче значения `null` строки не группируются. В рассматриваемом приложении группирование не выполняется, поэтому в метод передается значение `null`.
- ✓ **having.** Фильтр, описывающий группы строк, включаемых в результирующий набор (если строки группируются). При значении `null` включаются все группы строк. Если группировка не выполняется, необходимо передать значение `null` (как в данном примере).

- ✓ **orderBy.** Определение сортировки строк, отформатированное как директива `ORDER BY` без ключевых слов `ORDER BY`. При значении `null` применяется алгоритм сортировки, установленный по умолчанию, или строки не сортируются. В данном примере передается значение `null`, потому что порядок строк безразличен.
- ✓ **limit.** Ограничение количества возвращаемых строк с помощью директивы `LIMIT`. При значении `null` директива `LIMIT` не вставляется. В рассматриваемом приложении количество строк небольшое, и ограничивать его не имеет смысла, поэтому передается значение `null`.

Операция обновления

Метод обновления записей в безе данных принимает новые значения и заменяет ими существующие значения в заданных строках таблицы. Обновлять можно одну или много строк. Важно понимать, какие параметры необходимы методу `update()` и как они влияют на изменение записей в таблице базы данных. Метод `update()` принимает следующие параметры.

- ✓ **table.** Имя обновляемой таблицы. Находится в константе `DATABASE_TABLE`.
- ✓ **values.** Объект `ContentValues`, содержащий обновляемые поля. Используется переменная `args`, определенная в строке 64 листинга 12.4.
- ✓ **whereClause.** Директива `WHERE`, которая ограничивает обновляемые строки. В данном случае директива приказывает базе данных обновить строку с идентификатором `rowId`. Директива `WHERE` представляет собой строку `KEY_ROWID+"="_rowId`.
- ✓ **whereArgs.** Дополнительные аргументы директивы `WHERE`. В данном примере не используются, поэтому передается значение `null`.

Операция удаления

В методе `delete()` входные параметры используются для определения критерия удаления строк из таблицы. В зависимости от критерия метод может удалить произвольное количество строк, в том числе удалить все или ни одной строки. Это довольно опасный метод, потому что легко ошибочно удалить нужные записи. Поэтому важно понимать, как передаваемые параметры влияют на удаление данных. Метод `delete()` принимает следующие параметры.

- ✓ **table.** Имя таблицы, в которой нужно удалить строки. Значение имени находится в константе `DATABASE_TABLE`.
- ✓ **whereClause.** Необязательная директива `WHERE`, определяющая удаляемые строки. Если передать значение `null`, будут удалены все строки. В данном примере передается строковое значение, задающее удаление записи с идентификатором `rowId`.
- ✓ **whereArgs.** Дополнительные параметры директивы `WHERE`. В данном примере дополнительные параметры не используются, поэтому передается значение `null`.

Возврат всех задач с помощью курсора

Какая польза от созданной задачи, если она не видна в списке задач? Никакой. Следовательно, в список `ListView` деятельности `ReminderListActivity` нужно включить все задачи, хранящиеся в базе данных.

В листинге 12.5 приведен код деятельности `ReminderListActivity`, дополненный инструкциями чтения списка задач из базы данных и размещения списка в представлении `ListView`.

Листинг 12.5. Полный код деятельности `ReminderListActivity`, включая соединение с SQLite

```
public class ReminderListActivity extends ListActivity {
  private static final int ACTIVITY_CREATE=0;
  private static final int ACTIVITY_EDIT=1;

  private RemindersDbAdapter mDbHelper;                    5

  /** Вызывается при создании деятельности */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.reminder_list);
    mDbHelper = new RemindersDbAdapter(this);
    mDbHelper.open();
    fillData();                                           14
    registerForContextMenu(getListView());

  }

  private void fillData() {
    Cursor remindersCursor =
                mDbHelper.fetchAllReminders();            20
    startManagingCursor(remindersCursor);                 21

    // Создание массива полей заголовков
    String[] from = new
        String[]{RemindersDbAdapter.KEY_TITLE};           24

    // Создание массива полей, связанных с представлением
    int[] to = new int[]{R.id.text1};                     27

    // Создание адаптера курсора
    SimpleCursorAdapter reminders = new
       SimpleCursorAdapter(this, R.layout.reminder_row,
                remindersCursor, from, to);               30
    setListAdapter(reminders);                            31
  }

  // Код меню для краткости опущен
```

```
  @Override
  protected void onListItemClick(ListView l, View v,
                          int position, long id) {
    super.onListItemClick(l, v, position, id);
    Intent i = new
           Intent(this, ReminderEditActivity.class);
    i.putExtra(RemindersDbAdapter.KEY_ROWID, id);       40
    startActivityForResult(i, ACTIVITY_EDIT);
  }

  @Override
  protected void onActivityResult(int requestCode,
              int resultCode, Intent intent) {
    super.onActivityResult(requestCode,
              resultCode, intent); fillData();          48
  }

  @Override
  public boolean onContextItemSelected(
                               MenuItem item) {         52
    switch(item.getItemId()) {
      case R.id.menu_delete:
        AdapterContextMenuInfo info =
        (AdapterContextMenuInfo) item.getMenuInfo();    55
        mDbHelper.deleteReminder(info.id);              56
        fillData();                                     57
        return true;
    }
    return super.onContextItemSelected(item);
  }

}
```

Ниже приведено описание отмеченных строк кода, читающих список задач.

- ✓ **5.** Объявление переменной экземпляра `RemindersDbAdapter` на уровне класса. Экземпляр будет создан в методе `onCreate()`.
- ✓ **14.** Вызов метода `fillData()`, который загружает информацию из базы данных SQLite в представление `ListView`.
- ✓ **20.** В методе `fillData()` вызывается метод `fetchAllReminders()`, который извлекает все задачи из базы данных, как показано в строке 51 листинга 12.4.
- ✓ **21.** Вызов метода `startManagingCursor()`, определенного в классе `Activity`. Этот метод позволяет деятельности управлять жизненным циклом объекта `Cursor` на основе жизненного цикла деятельности. Например, когда деятельность принудительно останавливается, она автоматически вызывает метод `deactivate()` через объект `Cursor`. Когда деятельность вновь запускается, она вызывает метод `requery()`, вновь запуская этим объект `Cursor`. Когда деятельность уничтожается, все управляемые объекты `Cursor` автоматически закрываются.

- ✓ **24.** В этой строке кода определяется критерий выбора столбцов запросом. Задается выбор только данных столбца `TITLE`, т.е. названия задачи.
- ✓ **27.** Объявление массива представлений, которые нужно связать со строкой. При выводе названия задачи оно будет таким образом связано с идентификатором задачи. По этой причине в строке 24 переменная называется `from` (из), а в строке 27 — `to` (в). Значения строки 24 связываются со значениями строки 27.
- ✓ **30.** Создание адаптера курсора `SimpleCursorAdapter`, который связывает столбцы из объекта `Cursor` с представлениями `TextView`, определенными в файле компоновки XML. Таким образом, вы задаете, какой столбец и как должен быть отображен. Использование адаптера `SimpleCursorAdapter` и принимаемых параметров рассматривается далее.
- ✓ **31.** Объект `SippleCursorAdapter` передается как параметр методу `setListAdapter()`, чтобы проинформировать представление списка о том, где оно должно искать данные.
- ✓ **40.** Размещение идентификатора редактируемой задачи в намерении. Деятельность `ReminderEditActivity` просматривает намерение и, если находит идентификатор, предоставляет пользователю возможность отредактировать задачу.
- ✓ **48.** Вызов метода `fillData()`, когда деятельность возвращается из другой деятельности. Этот метод нужно вызвать, потому что пользователь мог обновить или добавить задачу. Вызов метода `fillData()` обеспечивает наличие задачи в представлении списка.
- ✓ **52.** Определение метода, который обработает событие контекстного меню, генерируемое, когда пользователь выбирает пункт контекстного меню после длинного щелчка на задаче в списке задач.
- ✓ **55.** Вызов метода `getMenuInfo()`, ассоциированного с пунктом, на котором выполнен щелчок с целью получения экземпляра `AdapterContextMenuInfo`. Этот класс предоставляет информацию о пункте меню, а также о пункте задачи в представлении списка, на которой был выполнен длинный щелчок.
- ✓ **56.** Обращение к классу `RemindersDbAdapter` для удаления задачи с идентификатором, извлеченным из поля `id` объекта `AdapterContextMenuInfo`. Поле `id` содержит идентификатор строки в представлении списка. Фактически этот идентификатор является идентификатором задачи `rowId` в базе данных.
- ✓ **57.** После удаления задачи из системы метод `fillData()` вызывается для повторного заполнения списка задач. При обновлении представления списка удаляется пункт, соответствовавший удаленной задаче.

Класс SimpleCursorAdapter

В строке 30 листинга 12.5 мы создали класс `SimpleCursorAdapter`. Рассмотрим подробнее, что делает каждый параметр его конструктора. Класс `SimpleCursorAdapter` выполняет много черновой работы вместо вас при связывании данных, возвращенных объектом `Cursor`, с представлением списка. При создании экземпляра класса `SimpleCursorAdapter` используются следующие параметры.

- ✓ **`this`.** Контекст, ассоциированный с адаптером.
- ✓ **`R.layout.reminder_row`.** Идентификатор ресурса компоновки, который определяет файл, используемый в данном пункте списка.
- ✓ **`reminderCursor`.** Курсор базы данных.
- ✓ **`from`.** Массив имен столбцов, используемых для связывания данных курсора с представлением. Этот массив определен в строке 24.
- ✓ **`to`.** Массив идентификаторов представления, которое должно отображать информацию о столбцах, заданных в параметре `from`. Этот массив определен в строке 27.
- ✓ Параметры `from` и `to` сообщают классу `SimpleCursorAdapter` о том, как он должен связать данные курсора с представлениями в компоновке строки.

Если теперь запустить приложение, будет виден список созданных задач. Задачи извлекаются из базы данных. Впрочем, пока что ни одной задачи не существует. Поэтому создайте одну или несколько задач, отобразив меню и выбрав пункт добавления новой задачи.

Удаление задачи

Чтобы удалить задачу, пользователь должен выполнить длинный щелчок на ней в списке представления `RemidnerListActivity` и выбрать в открывшемся контекстном меню команду удаления. С точки зрения программиста, для удаления задачи нужно вызвать метод `delete()` через объект базы данных SQLite. Этот метод вызывается в строке 48 листинга 12.4.

Метод `deleteReminder()` класса `RemindersDbAdapter` вызывается в методе `onContextSelectedItem()` в строке 56 листинга 12.5. Для удаления задачи из базы данных необходимо знать значение идентификатора задачи `rowId`. Для его получения используется объект `AdapterContextMenuInfo`, предоставляющий дополнительную информацию о меню. Эта информация передается контекстному меню, когда оно извлекается для представления `ListView`. Список загружается с курсором базы данных, поэтому объект `ListView` содержит значение `rowId`. В строке 55 листинга 12.5 мы получаем объект `AdapterContextMenuInfo`, а в строке 56 вызываем метод `delete()` и передаем ему значение `rowId` в качестве параметра. После этого вызывается метод `fillData()` для перерисовки задач на экране. После перерисовки список готов к работе: пользователь вновь может создавать, читать, обновлять и удалять задачи.

Обновление задачи

Для пользователя процесс обновления довольно простой, однако, с точки зрения программиста, это довольно сложный процесс, потому что для обновления и создания задачи применяется одна и та же деятельность. Следовательно, в код должна быть включена проверка, какая операция сейчас выполняется: редактирование существующей задачи или создание новой.

Проверка выполняется на основе намерения, используемого для запуска деятельности. При щелчке на задаче в деятельности `ReminderListActivity` запускается следующая деятельность.

```
Intent i = new Intent(this, ReminderEditActivity.class);
i.putExtra(RemindersDbAdapter.KEY_ROWID, id);
startActivityForResult(i, ACTIVITY_EDIT);
```

Этот код приказывает операционной системе запустить деятельность `ReminderEditActivity` с параметром `i` класса `Intent`, содержащим дополнительную информацию — идентификатор строки задачи (`id`), которую нужно редактировать. На стороне объекта `ReminderEditActivity` выполняется проверка, содержит ли намерение параметр `id`. Если это так, то выполняется операция редактирования, и в форму загружается информация о задаче. Пользователь редактирует в форме информацию о задаче. Если же параметра `id` в намерении нет (это происходит, когда пользователь выбрал в меню добавление новой задачи), приложение предоставляет пользователю пустую форму новой задачи.

Описанная выше процедура представлена в листинге 12.6. Полужирным шрифтом отмечен новый код.

Листинг 12.6. Деятельность `ReminderEditActivity`, поддерживающая вставку и обновление задач

```
public class ReminderEditActivity extends Activity {

  // Здесь объявлены другие переменные уровня класса,
  // для краткости они опущены
  private Long mRowId;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mDbHelper = new RemindersDbAdapter(this);
    setContentView(R.layout.reminder_edit);
    mCalendar = Calendar.getInstance();
    mTitleText = (EditText) findViewById(R.id.title);
    mBodyText = (EditText) findViewById(R.id.body);
    mDateButton = (Button) findViewById(
                                  R.id.reminder_date);
    mTimeButton = (Button) findViewById(
                                  R.id.reminder_time);
    mConfirmButton = (Button) findViewById(
                                  R.id.confirm);
```

```
  mRowId = savedInstanceState != null                    22
    ? savedInstanceState.getLong(
                    RemindersDbAdapter.KEY_ROWID)
    : null;
  registerButtonListenersAndSetDefaultText();
}

private void setRowIdFromIntent() {                      28
  if (mRowId == null) {
    Bundle extras = getIntent().getExtras();
    mRowId = extras != null
      ? extras.getLong(RemindersDbAdapter.KEY_ROWID)
      : null;
  }
}

@Override
protected void onPause() {
  super.onPause();
  mDbHelper.close();                                     40
}

@Override
protected void onResume() {                              44
  super.onResume();
  mDbHelper.open();                                      46
  setRowIdFromIntent();                                  47
  populateFields();                                      48
}

// Для краткости здесь опущены следующие компоненты:
// элемент выбора даты, обработчик щелчка на кнопке,
// процедура обновления текста кнопки и процедура
// создания диалогового окна

private void populateFields() {                          55
  if (mRowId != null) {
    Cursor reminder =
              mDbHelper.fetchReminder(mRowId);           57
    startManagingCursor(reminder);                       58
    mTitleText.setText(reminder.getString(
    reminder.getColumnIndexOrThrow(
          RemindersDbAdapter.KEY_TITLE)));               60
    mBodyText.setText(reminder.getString(
    reminder.getColumnIndexOrThrow(
          RemindersDbAdapter.KEY_BODY)));                61
    SimpleDateFormat dateTimeFormat =
          new SimpleDateFormat(DATE_TIME_FORMAT)         63
    Date date = null;                                    64
    try {
      String dateString = reminder.getString(
```

```
            reminder.getColumnIndexOrThrow(
            RemindersDbAdapter.KEY_DATE_TIME));          67
          date = dateTimeFormat.parse(dateString);       68
          mCalendar.setTime(date);                       69
        } catch (ParseException e) {                     70
          Log.e("ReminderEditActivity",
                e.getMessage(), e);                      71
        }
      }

      updateDateButtonText();
      updateTimeButtonText();
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
      super.onSaveInstanceState(outState);
      outState.putLong(RemindersDbAdapter.KEY_ROWID,
                                          mRowId);       82
    }

    private void saveState() {
      String title = mTitleText.getText().toString();
      String body = mBodyText.getText().toString();

      SimpleDateFormat dateTimeFormat = new
        SimpleDateFormat(DATE_TIME_FORMAT);
      String reminderDateTime =
        dateTimeFormat.format(mCalendar.getTime());

      if (mRowId == null) {                              94
        long id = mDbHelper.createReminder(title,
                  body, reminderDateTime);               95
        if (id > 0) {                                    96
           mRowId = id;                                  97
        }
      } else {
        mDbHelper
        .updateReminder(mRowId, title, body,
                              reminderDateTime);        100
      }
    }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **22.** Проверка, содержит ли сохраненное состояние класса идентификатор `mRowId`. Состояние устанавливается в методе `onSaveInstanceState()`.
- ✓ **28.** Извлечение объекта `mRowId` из намерения, которое запустило деятельность. Если объект намерения `Intent` не содержит дополнительной информации, объект `mRowId` остается равным `null`. Обратите

внимание на использование типа `Long` с прописной буквой `L`. Это ссылочное значение `long`, которое может быть равным `null` или содержать значение типа `long`.

- ✓ **40.** Закрытие базы данных при завершении деятельности или ее переходе в режим паузы.
- ✓ **44.** Вызов метода `onResume()` в жизненном цикле деятельности (см. рис. 5.1).
- ✓ **46.** Открытие базы данных, чтобы ее можно было использовать в деятельности.
- ✓ **47.** Вызов метода, получающего объект `mRowId` из намерения, которое запустило данную деятельность.
- ✓ **48.** Метод `populateFields()` заполняет поля формы.
- ✓ **55.** Заголовок определения метода, который заполняет поля формы.
- ✓ **57.** Извлечение объекта `Cursor` из базы данных SQLite с помощью идентификатора `mRowId`. Метод `fetchReminder()` определяется в строке 55 листинга 12.4.
- ✓ **58.** Запуск управления деятельностью в объекте `Cursor`.
- ✓ **60.** Установка названия задачи с помощью объекта `Cursor`. Чтобы извлечь значения из курсора, нужно знать индекс столбца в курсоре. Метод `getColumnInderOrThrow()`, вызванный через объект курсора, извлекает индекс столбца на основе его имени. Имея индекс столбца, можно получить его значение, вызвав метод `getString()` и передав ему индекс столбца в качестве параметра. Получив значение столбца, деятельность устанавливает текст названия в представлении `EditText`, хранящемся в переменной `mTitleText`.
- ✓ **61.** Извлечение и установка текста задачи `mBodyText` для представления `EditText` с помощью того же метода, что и в строке 60, но теперь используются другие индекс и имя столбца.
- ✓ **63.** База данных SQLite не поддерживает тип даты, поэтому дата и время хранятся в строковом формате. Для создания строки на основе типа даты в Java используется класс `SimpleDateFormat`. Он принимает дату в формате Java и создает строку с отформатированным представлением даты.
- ✓ **64.** Создание экземпляра объекта даты типа `Date` с помощью пакета `java.util.Date`.
- ✓ **67.** Извлечение строки с датой из курсора.
- ✓ **68.** Запись даты и времени в объект `Date`.
- ✓ **69.** Запись даты и времени в объект `Calendar`. В библиотеке Java объект `Calendar` более мощный, чем `Date`, поэтому последний используется только для преобразования строки в объект `Calendar`. Эту же операцию можно было выполнить с помощью методов класса `Calendar`.

- ✓ **70.** Перехват исключений, которые могут возникнуть вследствие некорректного формата даты в строке `SimpleDateFormat`. Используемое в данном блоке исключение `ParseException` определено в пакете `java.text.ParseException`.
- ✓ **71.** Запись сообщения об ошибке в системный журнал.
- ✓ **82.** Сохранение объекта `mRowId` в состоянии приложения. Чтобы можно было извлечь и сохранить состояние в `Bundle` на уровне деятельности, вызывается метод `onSaveInstanceState()`. Вызов происходит до уничтожения деятельности, чтобы при возобновлении деятельности ее можно было привести в правильное состояние, как в методе `onResume()`. В строке 22, до проверки входных данных намерения, выполняется проверка того, присутствует ли идентификатор строки в объекте `savedInstanceState`. Это необходимо, поскольку Android может уничтожить деятельность по причинам, независимым от приложения. Это может произойти вследствие входного телефонного звонка, воспроизведения музыки или применения средств Google Map, если для этих приложений недостаточно ресурсов мобильного устройства. Позже, при возврате к приложению Task Reminder, на основе проверки объекта `savedInstanceState` принимается решение, можно ли возобновить деятельность. Сохранение объекта `mRowId` в этом объекте позволяет возобновить деятельность, начиная с предыдущего состояния.
- ✓ **94.** В методе `saveState()` необходимо выяснить, какой процесс выполняется: сохранение новой задачи или обновление существующей. Если переменная `mRowId` равна `null`, значит, идентификатор можно найти в объекте `savedInstanceState` или во входном намерении. Следовательно, задача считается новой.
- ✓ **95.** В базе данных создается новая задача.
- ✓ **96.** Идентификатор задачи, возвращенный базой данных, должен быть больше нуля.
- ✓ **97.** Присвоение переменной `mRowId` идентификатора, полученного от базы данных.
- ✓ **100.** Обновление задачи. Идентификатор передается для обновления названия, текста, даты и времени конкретной задачи.

Сейчас при запуске приложения в эмуляторе можно создавать, читать, обновлять и удалять задачи. Осталось лишь создать строку состояния и добавить в нее уведомления о задачах. Этим мы займемся в главе 14.

Глава 13

Класс менеджера сигналов

В этой главе...

- Зачем нужен класс `AlarmManager`
- Запуск процесса с помощью `AlarmManager`
- Перезагрузка устройства

В повседневной жизни многие дела нужно делать ежедневно по расписанию. Утром вы встаете, умываетесь, завтракаете, собираетесь на работу и т.д. В этой цепочке дел самая сложная задача — проснуться. Главным образом не потому, что вы лентяй, а по той причине, что в организме человека нет встроенного будильника, который утром разбудит его в нужный момент времени. Следовательно, для успешного решения данной задачи (т.е. чтобы не опаздывать на работу) вам необходимо специальное механическое или электронное устройство — будильник.

Нечто вроде будильника необходимо и для нашего приложения Task Reminder. Предположим, вы предоставили пользователю возможность создать список задач, и приложение отображает их на экране. Однако, чтобы устройство напомнило пользователю о некоторой задаче в нужный момент времени, что-то должно "подтолкнуть" приложение в этот момент, другими словами — инициировать процесс уведомления пользователя. В операционной системе Windows 7 для этого существует служба, которая называется "Планировщик заданий". В Linux для этого предназначена команда `cron`. Платформа Android для генерации сигнала в заданный момент времени предоставляет класс `AlarmManager`, играющий ту же роль, что и планировщик заданий в Windows.

Зачем нужен класс AlarmManager

При работе с приложением Task Reminder пользователю нужно предоставить возможность установить следующие параметры задачи: название, описание и время напоминания. Чтобы приложение напомнило пользователю о задаче, ему самому что-то должно напомнить о ней. Рассмотрим следующий сценарий. Вы добавили в приложение Task Reminder несколько задач, о которых оно должно напомнить вам, предположим, в течение сегодняшнего дня. Вы положили устройство в карман и занялись другими делами. Если устройство не напомнит вам о необходимости выполнить указанные задачи, вы, вполне вероятно, забудете о них или вспомните, когда будет поздно. Следовательно, приложение в определенный момент должно не просто зазвонить, а известить вас о конкретной задаче. Но что заставит приложение "проснуться" в заданный момент времени? Класс `AlarmManager`.

С помощью класса `AlarmManager` можно программно составить расписание моментов времени, когда приложению будут подаваться определенные сигналы. Когда

наступает заданный момент, класс AlarmManager автоматически генерирует широковещательное намерение, передаваемое операционной системой всем приложениям. Однако отреагировать на него может только одно приложение — Task Reminder. В качестве реакции на широковещательное намерение можно запрограммировать все, что может сделать мобильное устройство, — вывод окна, воспроизведение видеоклипа, отправку электронного письма, звуковой сигнал, запуск рингтона, отображение уведомления в строке состояния и т.д. В главе 14 рассматривается оповещение пользователя с помощью строки состояния.

Приложение Task Reminder удерживает процессор в "бодрствующем состоянии", пока выполняется метод onReceive(). Это гарантирует, что телефон не переключится в спящий режим, пока приложение не закончит обрабатывать широковещательное намерение. В противном случае приложение, прерванное "на полуслове", может испортить базу данных. Вот почему нам понадобилось разрешение WAKE_LOCK (блокировка отключения процессора), которое мы рассмотрели в предыдущей главе.

Запуск процесса с помощью объекта AlarmManager

Чтобы с помощью класса AlarmManager запустить некоторый процесс, нужно сначала установить и сконфигурировать сигнал оповещения. В приложении Task Reminder лучше всего это сделать непосредственно после сохранения задачи в методе saveState(). Однако, прежде чем приступить к кодированию, нужно добавить в проект четыре класса.

- ✓ **ReminderManager.java.** Этот класс отвечает за конфигурирование напоминаний с помощью класса AlarmManager.
- ✓ **OnAlarmReceiver.java.** В этом классе обрабатываются широковещательные сообщения, передаваемые при генерации сигналов классом AlarmManager. В файл манифеста приложения AndroidManifest.xml необходимо добавить приведенный ниже код.

  ```
  <receiver android:name=".OnAlarmReceiver" />
  ```

  Точка перед OnAlarmReceiver информирует платформу Android о том, что приемник широковещательных сообщений, определенный в файле AndroidManifest.xml, находится в текущем пакете.
- ✓ **WakeReminderIntentService.java.** Этот абстрактный класс отвечает за получение и отмену блокировки отключения процессора.
- ✓ **ReminderService.java.** Данный класс является реализацией базового класса WakeReminderIntentService, который обрабатывает создание уведомления (см. главу 14).

  Чтобы приложение распознавало службу WakeReminderIntentService, в элемент приложения в файле AndroidManifest.xml нужно добавить следующий код:

  ```
  <service android:name=".ReminderService" />
  ```

Создание класса ReminderManager

Как сказано выше, класс ReminderManager отвечает за установку расписания сигналов операционной системы с помощью класса AlarmManager. Для достижения высокого уровня абстракции в классе ReminderManager выполняются все действия, имеющие отношение к установке расписания в классе AlarmManager.

Добавьте приведенный ниже код в конец метода saveState() класса ReminderEditActivity, чтобы добавить в расписание сигнал о данной задаче.

```
new ReminderManager(this).setReminder(mRowId, mCalendar);
```

Эта строка приказывает классу ReminderManager установить в объекте AlarmManager для задачи с идентификатором mRowId новое напоминание в момент времени, определенный в переменной mCalendar.

В листинге 13.1 приведен код класса ReminderManager.

Листинг 13.1. Класс ReminderManager

```
public class ReminderManager {

private Context mContext;
private AlarmManager mAlarmManager;
public ReminderManager(Context context) {                6
  mContext = context;
  mAlarmManager =
    (AlarmManager)context.getSystemService(
                          Context.ALARM_SERVICE);        9
}

public void setReminder(Long taskId,
                        Calendar when) {                 12
  Intent i =
        new Intent(mContext, OnAlarmReceiver.class);     13
  i.putExtra(RemindersDbAdapter.KEY_ROWID,
                                 (long)taskId);          14

  PendingIntent pi =
    PendingIntent.getBroadcast(mContext, 0, i,
    PendingIntent.FLAG_ONE_SHOT);                        16

  mAlarmManager.set(AlarmManager.RTC_WAKEUP,
     when.getTimeInMillis(), pi);                        17
    }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **6.** Заголовок конструктора класса ReminderManager. В качестве параметра конструктор принимает объект context.
- ✓ **9.** Получение объекта AlarmManager путем вызова метода getSystemService().

- ✓ **12.** Заголовок определения метода `setReminder()`. Метод принимает идентификатор задачи в базе данных и объект `Calendar`, определяющий момент подачи сигнала.
- ✓ **13.** Создание объекта намерения, отвечающего за реакцию приложения на сигнал. В данном случае намерение попадает в приемник `OnAlarmReceiver`.
- ✓ **14.** Объекту намерения предоставляется дополнительная информация — идентификатор задачи в базе данных.
- ✓ **16.** Класс `AlarmManager` работает в отдельном процессе, поэтому для того, чтобы он известил приложение о необходимости выполнить определенную операцию, нужно создать объект отложенного намерения класса `PendingIntent`. Этот объект содержит объект `Intent`, созданный в строке 13. Объекту `PendingIntent` передается флажок `FLAG_ONE_SHOT`, означающий, что он может быть использован только один раз.
- ✓ **17.** Вызов метода `set()` класса `AlarmManager` для установки сигнала в расписание. Метод `set()` принимает следующие параметры.
  - **`type: AlarmManager.RTC_WAKEUP`.** Тип операции — пробуждение устройства в момент, заданный параметром `triggerAtTime`.
  - **`triggerAtTime: when.getTimeInMillis()`.** Время подачи сигнала. Объект `Calendar` предоставляет метод `getTimeInMillis()`, который преобразует значение даты и времени в тип `long`. Значение типа `long` равно количеству миллисекунд, прошедшему с момента полуночи 1 января 1970 года по Гринвичу (тем не менее, в объекте `Calendar` время задается не по Гринвичу, а в локальном часовом поясе).
  - **`operation: pi`.** Отложенное намерение, запускаемое при подаче сигнала.

Если сигнал уже вставлен в расписание с отложенным намерением, имеющим ту же сигнатуру, предыдущий сигнал будет удален, а новый вставлен.

Создание класса `OnAlarmReceiver`

Класс `OnAlarmReceiver` (листинг 13.2) отвечает за обработку намерения, запускаемого при подаче сигнала. Фактически класс подключен к системе подачи сигнала, потому что он является простой реализацией базового класса широковещательного приемника `BroadcastReceiver`, который реагирует на широковещательные события в операционной системе.

Листинг 13.2. Класс `OnAlarmReceiver`

```
public class OnAlarmReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    long rowid =
      intent.getExtras().getLong(
                        RemindersDbAdapter.KEY_ROWID);    4
      WakeReminderIntentService.
```

```
                acquireStaticLock(context);                    6
        Intent i = new Intent(context,
                        ReminderService.class);                8
        i.putExtra(RemindersDbAdapter.KEY_ROWID, rowid);       9
        context.startService(i);                              10
    }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **4.** Извлечение идентификатора задачи, хранящегося в базе данных, из намерения, после того как приемник начал обрабатывать намерение.
- ✓ **6.** Служба `WakeReminderIntentService` запрашивает статическую блокировку отключения процессора, чтобы устройство работало до тех пор, пока не будет завершена обработка намерения.
- ✓ **8.** Создание нового объекта намерения, которое запустит класс `ReminderService`, уведомляющий пользователя о задаче.
- ✓ **9.** Запись идентификатора задачи в намерение, используемое для запуска службы уведомления пользователя. Служба `ReminderService` получает необходимый ей идентификатор задачи.
- ✓ **10.** Запуск службы `ReminderService` через текущий контекст.

Это первая установленная нами точка входа для сигнала. Во время работы приемника `BroadcastReceiver` устройство может отключиться по независимым от приложения причинам. Это очень нежелательно, потому что обработка задачи останется незавершенной, записи в базе данных будут испорчены и приложение останется в нерабочем состоянии.

Когда генерируется сигнал, Android запускает отложенное намерение, вставленное в расписание с данным сигналом. Намерение будет принято и обработано любым широковещательным приемником, предназначенным для его обработки.

Это ваш второй "набег" на объект `BroadcastReceiver`, однако я все еще не рассказал вам, как он работает. Роль этого компонента сводится к получению широковещательных сообщений и реагированию на них. Никакие элементы пользовательского интерфейса он не отображает, а лишь запускает деятельность в ответ на полученное сообщение. Объект `OnAlarmReceiver` является экземпляром базового класса `BroadcastReceiver`.

Когда `AlarmManager` генерирует широковещательное отложенное намерение, на него реагирует класс `OnAlarmReceiver`, потому что оно адресовано ему, что видно из строки 13 листинга 13.1. Этот класс принимает намерение, блокирует процессор и выполняет необходимые операции.

Создание класса `WakeReminderIntentService`

Этот класс (его код приведен в листинге 13.3) управляет блокировкой процессора. Блокировка обеспечивает активность процессора (но не обязательно экрана), пока не будет выполнена некоторая работа, которую нельзя бросить на полпути. После этого класс `WakeReminderIntentService` освобождает блокировку, и операционная система может делать с процессором все, что ей нужно, например переключить в спящий режим.

Листинг 13.3. Класс `WakeReminderIntentService`

```
public abstract class WakeReminderIntentService
      extends IntentService { abstract void
          doReminderWork(Intent intent);                  2

  public static final String
    LOCK_NAME_STATIC=
        "com.dummies.android.taskreminder.Static";        3
  private static PowerManager.WakeLock
                            lockStatic=null;              4

  public static void acquireStaticLock(
          Context context) {getLock(context).acquire(); 5
  }

    synchronized private static PowerManager.WakeLock
        getLock(Context context) {                        8
    if (lockStatic==null) {
      PowerManager mgr=(PowerManager)context
           .getSystemService(Context.POWER_SERVICE);     10

    lockStatic=mgr.newWakeLock(PowerManager.
      PARTIAL_WAKE_LOCK, LOCK_NAME_STATIC);               12
    lockStatic.setReferenceCounted(true);                 13
  }
    return(lockStatic);                                   15
  }

  public WakeReminderIntentService(String name) {         18
    super(name);
  }

  @Override
  final protected void onHandleIntent(
                           Intent intent) {               23
    try {
        doReminderWork(intent);                           25
    } finally {
       getLock(this).release();                           27
    }

  }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **2.** Этот абстрактный метод реализован в каждом дочернем классе, например в классе `ReminderService`, что видно в строке 7 листинга 13.4.
- ✓ **3.** Имя дескриптора блокировки, используемое для связывания процессора. Используется при отладке.

- ✓ **4.** Закрытая статическая переменная блокировки. Ссылка на нее есть далее в классе.
- ✓ **5.** Вызов метода получения блокировки, объявленного в строке 8. После возврата вызова `getLock()` вызывается метод `aquire()`, чтобы обеспечить требуемое состояние устройства, а именно — состояние частичной блокировки. Эта блокировка не дает устройству перейти в спящий режим, но не влияет на отключение экрана.
- ✓ **8.** Определение метода `getLock()`, который возвращает блокировку `PowerManager.WakeLock`, позволяющую запретить платформе Android переключать устройство в спящий режим, пока не будут выполнены определенные операции.
- ✓ **10.** Извлечение класса `PowerManager` путем вызова метода `getSystemService()`. Этот класс используется для создания блокировки.
- ✓ **12.** Создание блокировки `WakeLock` путем вызова метода `newWakeLock()`. Метод принимает следующие параметры.
  - **`flags: PowerManager.PARTIAL_WAKE_LOCK`.** Вызову можно предоставить много дескрипторов, однако в данном случае используется только один дескриптор `PARTIAL_WAKE_LOCK`. Он информирует Android о том, что процессор должен быть включен, однако обязательного включения экрана не требуется.
  - **`tag: LOCK_NAME_STATIC`.** Строковая константа с именем дескриптора блокировки, используемая для отладки. Определена в строке 3.
- ✓ **13.** Эта строка информирует `PowerManager` о том, что ссылка была учтена.
- ✓ **15.** Возвращение объекта блокировки `WakeLock` вызывающему методу.
- ✓ **18.** Конструктор класса `WakeReminderIntentService`, используемый только для отладки приложения.
- ✓ **23.** Заголовок определения метода `onHandleIntent`, вызываемого через объект `IntentService`. Как только служба запускается, метод `onHandleIntent()` вызывается для обработки намерения, полученного службой.
- ✓ **25.** Служба вызывает метод, выполняющий все необходимые операции по напоминанию пользователю о задаче.
- ✓ **27.** Независимо от успешности вызова метода `doReminderWork()` необходимо снять блокировку `WakeLock`. Если не сделать этого, устройство останется во включенном состоянии, пока не будет перезагружено. Это весьма нежелательный режим, потому что он приводит к быстрой разрядке батареи. Поэтому в завершающем блоке `finally` инструкции `try-catch` вызывается метод `release()`. Блок `finally` выполняется всегда, независимо от того, было ли сгенерировано исключение в блоке `try`.

Пока что в классе `ReminderService` нет ни одной реализации метода `doReminderWork()`, тем не менее приложение Task Reminder реагирует на сигналы. Попробуйте создать несколько задач и установить точки прерывания в режиме отладки, чтобы просмотреть маршрут выполнения в методе `doReminderWork()` класса `ReminderService`.

Класс `AlarmManager` не сохраняет расписание сигналов. Это означает, что при перезагрузке устройства расписание уничтожается. Если мы хотим, чтобы приложение напоминало пользователю о задачах после перезагрузки, то должны вновь создать расписание в `AlarmManager`.

Предыдущий код демонстрирует все, что необходимо для блокировки отключения процессора в неподходящий момент. Код запрашивает блокировку перехода в спящий режим и не снимает ее, пока не будет завершен вызов метода `doReminderWork()`.

Создание класса `ReminderService`

Класс `ReminderService` (листинг 13.4) отвечает за выполнение всех необходимых операций при получении сигнала. Однако реализация класса, приведенная в данной главе, всего лишь создает оболочку для необходимых процедур. Фактически нужно запрограммировать вывод уведомления для пользователя в строке состояния в ответ на сигнал класса `AlarmManager`. Это будет сделано в главе 14.

Листинг 13.4. Класс `ReminderService`

```
public class ReminderService extends
                          WakeReminderIntentService {   1
  public ReminderService() {
    super("ReminderService");
  }

  @Override
  void doReminderWork(Intent intent) {                  7
    Long rowId = intent.getExtras()
      .getLong(RemindersDbAdapter.KEY_ROWID);           8
    // Здесь нужно вставить код для строки состояния
  }
}
```

Ниже приведено описание отмеченных строк.

- ✓ **1.** Данный класс наследует класс `WakeReminderIntentService`.
- ✓ **7.** Здесь реализуется абстрактный метод `doReminderWork()` класса `WakeReminderIntentService`.
- ✓ **8.** Извлечение идентификатора задачи из объекта намерения, переданного в данный класс.

Как сказано выше, данный класс всего лишь извлекает идентификатор задачи из намерения.

Перезагрузка устройства

Я вполне допускаю, что после длинного хлопотливого дня и хорошего ночного отдыха я часто забываю многие вещи. Ведь я всего лишь человек, а не компьютер. Чаще всего мне нужно напомнить об определенных задачах утром, когда я просыпаюсь. Этим я похож на класс `AlarmManager`, который тоже забывает о расписании сигналов при выключении устройства. Ему тоже нужно напомнить о них при каждой перезагрузке. Эта работа не такая уж сложная, однако она должна быть сделана.

Если не установить расписание сигналов повторно, они не будут генерироваться, потому что после перезагрузки их нет в Android.

Создание приемника загрузки

В предыдущей главе мы объявили в манифесте разрешение `RECEIVE_BOOT_COMPLETED`. Оно позволяет приложению получать от операционной системы Android широковещательное извещение о том, что устройство перезагружено и готово взаимодействовать с пользователем. Операционная система Android может сгенерировать широковещательное сообщение, когда перезагрузка завершается, поэтому в проект нужно добавить еще один объект `BroadcastReceiver`. Этот приемник широковещательных извещений отвечает за обработку извещения о перезагрузке Android. Когда извещение получено, приемник устанавливает соединение с базой данных SQLite посредством адаптера `RemindersDbAdapter`, проходит по списку задач и восстанавливает сигналы для каждой задачи. Благодаря этим операциям сигналы не теряются при перезагрузке.

Добавим в приложение новый объект `BroadcastReceiver`. В приложении Task Reminder этот объект имеет имя `onBootReceiver`. Нужно также добавить в манифест приложения `AndroidManifest.xml` следующий код.

```
<receiver android:name=".OnBootReceiver">
  <intent-filter>
    <action android:name=
              "android.intent.action.BOOT_COMPLETED" />
  </intent-filter>
</receiver>
```

Этот код информирует Android о том, что объект `OnBootReceiver` должен получать извещения о загрузке для действия `BOOT_COMPLETED`. Попросту говоря, объект `OnBootReceiver` должен узнать о том, что устройство перезагружено.

Полный код класса `OnBootReceiver` приведен в листинге 13.5.

Листинг 13.5. Класс `OnBootReceiver`

```
public class OnBootReceiver
               extends BroadcastReceiver {              1

  @Override
  public void onReceive(Context context,
                        Intent intent) {              4
    ReminderManager reminderMgr =
                  new  ReminderManager(context);      6
    RemindersDbAdapter dbHelper =
```

```
                new RemindersDbAdapter(context);
    dbHelper.open();
    Cursor cursor = dbHelper.fetchAllReminders();      11
    if(cursor != null) {
      cursor.moveToFirst();                            14
      int rowIdColumnIndex = cursor.getColumnIndex(
                  RemindersDbAdapter.KEY_ROWID);
      int dateTimeColumnIndex =
      cursor.getColumnIndex(
                RemindersDbAdapter.KEY_DATE_TIME);
      while(cursor.isAfterLast() == false) {           19
        Long rowId = cursor.getLong(rowIdColumnIndex);
        String dateTime =
                 cursor.getString(dateTimeColumnIndex);
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat format = new
              SimpleDateFormat(
                 ReminderEditActivity.DATE_TIME_FORMAT);

        try {
          java.util.Date date = format.parse(dateTime);  27
          cal.setTime(date);                             28

          reminderMgr.setReminder(rowId, cal);           30
        } catch (ParseException e) {
          Log.e("OnBootReceiver", e.getMessage(), e);    32
        }
        cursor.moveToNext();                             35
      }
        cursor.close() ;                                 37
      }
        dbHelper.close();                                40
    }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Заголовок определения класса `OnBootReceiver`.
- ✓ **4.** Заголовок метода `onReceive()`, вызываемого, когда приемник получает намерение.
- ✓ **6.** Создание объекта `ReminderManager`, позволяющего создать расписание сигналов.
- ✓ **11.** Получение курсора со всеми задачами из адаптера базы данных `RemindersDbAdapter`. Этот же вызов используется для загрузки объекта `ListView` в объект `ReminderListActivity`.
- ✓ **14.** Переход к первой записи объекта `Cursor`. Курсор может содержать много записей; проход по записям начинается с первой.
- ✓ **19.** Цикл `while`, проверяющий, остались ли еще записи. Если переменная цикла не равна `true`, значит, последней записи мы еще не достигли

и можно переходить к следующей записи в строке 35. Значение `true` сигнализирует о том, что в курсоре записей больше нет.

- ✓ **27.** Чтение даты из строки, извлеченной из базы данных.
- ✓ **28.** Обновление объекта `Calendar` значением, полученным из курсора базы данных. В этой строке объект `Date` преобразуется в объект `Calendar`.
- ✓ **30.** Запись в расписание новой задачи с идентификатором, полученным из базы данных, и временем, хранящимся в объекте `Calendar`.
- ✓ **32.** Запись исключений в системный журнал. Эта инструкция используется при отладке приложения. Во время эксплуатации никаких исключений не должно быть.
- ✓ **35.** Переход к следующей записи в курсоре. Если в курсоре больше нет записей, метод `isAfterLast()` возвращает `true`, и программа выходит из цикла `while`. В противном случае управление вновь передается строке 19, в результате чего класс обрабатывает следующую запись.
- ✓ **37.** Закрытие курсора, поскольку он больше не нужен. Возможно, вы обратили внимание на то, что в прошлый раз, когда мы работали с объектом `Cursor`, мы никогда не закрывали его. В этом не было необходимости, потому что объект деятельности управляет курсором. Сейчас же управление принадлежит широковещательному приемнику, поэтому доступа к объекту деятельности нет.
- ✓ **40.** Закрытие адаптера `ReminderDbAdapter`, который, в свою очередь, закрывает базу данных. Они больше не нужны.

Если сейчас запустить приложение, создать несколько задач и перезагрузить устройство, расписание сигналов останется в операционной системе. Если вы отлаживаете приложение, не забудьте присвоить в манифесте атрибуту `debuggable` значение `true`, иначе в рабочей среде Eclipse вы не найдете многих средств отладки.

Проверка приемника загрузки

Чтобы тщательнее проверить работу приемника загрузки `OnBootReceiver`, поместите в цикл `while` инструкции записи сообщений в системный журнал.

```
Log.d("OnBootReceiver",
                "Добавление сигнала при загрузке.");
Log.d("OnBootReceiver",
                "Индекс столбца = "+rowIdColumnIndex);
```

Эти сообщения можно увидеть в системном журнале, открыв окно DDMS. Закройте эмулятор (или выключите физическое устройство) и опять запустите его. Посмотрите на поток сообщений в DDMS и найдите сообщения для объекта `OnBootReceiver`. Если в базе данных есть две задачи, вы увидите два набора сообщений, информирующих о том, что при загрузке в расписание были добавлены два сигнала. Второе сообщение содержит индекс столбца, содержащего идентификаторы задач.

Глава 14

Обновление строки состояния

В этой главе...

- Инфраструктура строки состояния
- Использование менеджера уведомлений
- Изменение уведомления
- Удаление уведомлений

В предыдущих главах я неоднократно упоминал о различных способах привлечения внимания пользователя, включая активизацию диалоговых окон, уведомлений и новых деятельностей. Все это хорошо в соответствующих ситуациях, однако в нашем случае нежелательно отвлекать внимание пользователя от его текущего занятия. Следовательно, нужен способ проинформировать пользователя о том, что он должен уделить внимание некоторой задаче, когда у него будет время. Для этого предназначена строка состояния Android.

Структура строки состояния

Как известно, один рисунок стоит тысячи слов, однако без слов тоже не обойтись. Поэтому рассмотрим в качестве примера строку состояния, показанную на рис. 14.1.

Значки строки состояния

На рис. 14.1 первый слева значок в левом верхнем углу — это уведомление календаря, информирующее меня о предстоящей встрече с коллегами сегодня. Второй значок сообщает о том, что устройство подключено к другому устройству (компьютеру) посредством порта USB, а третий значок говорит об активном режиме отладки через USB. Пользователь может перетащить строку состояния вниз, в результате чего будет отображена дополнительная информация об этих трех уведомлениях (рис. 14.2).

На рис. 14.2 видно, что с каждым значком ассоциировано расширенное представление, отображающее дополнительную информацию о значке. Пользователь может выбрать расширенное представление, в результате чего будет запущено связанное с ним действие.

Использование строки состояния для уведомления пользователя

Строка состояния предоставляет разные инструменты уведомления пользователя. Для этого предназначены не только простые значки, отображаемые в верхней части экрана. Вы можете расширить уведомление, установив дополнительные флажки. Ниже перечислены доступные инструменты уведомления.

Рис. 14.1. Строка состояния с несколькими значками

Рис. 14.2. Строка состояния после перетаскивания вниз

- ✓ **Вибрация.** При получении уведомления можно включить вибрацию устройства. Этот инструмент полезен, когда устройство находится в кармане пользователя.
- ✓ **Звук.** При поступлении уведомления можно включить воспроизведение рингтона или любого музыкального файла, встроенного в приложение.
- ✓ **Вспышка.** Многие устройства содержат встроенный светодиодный источник яркого света, к которому Android предоставляет программный доступ. Можно задать яркость, периодичность и цвет вспышки. Если источник поддерживает только один цвет (например, белый), то цвет, заданный программно, будет игнорироваться.

Добавление различных инструментов уведомления расширяет ваш арсенал и делает приложение более полезным для пользователя, потому что в разных ситуациях уместны разные способы уведомления.

Строка состояния — мощный инструмент, который можно использовать для предоставления пользователю информации о жизненном цикле приложения. Значки, вибрация, свет и звук — очень полезные средства, но строка состояния предоставляет еще более полезную возможность — развертывание строки состояния, в результате чего отображается дополнительная информация об уведомлениях.

Строку состояния можно использовать для информирования пользователя о различных процессах, таких как получение электронных писем, процент выполнения длительной задачи, отображение состояния устройства и пр. На рис. 14.3 показан индикатор прогресса, информирующий пользователя о процессе загрузки из Интернета приложения Foursquare.

Рис. 14.3. Загружено 60% инсталляционного файла Foursquare

Как разработчик, вы имеете программный доступ к пользовательским расширенным представлениям, т.е. представлениям, отображаемым, когда пользователь перетаскивает строку состояния вниз, разворачивая ее таким образом.

Использование менеджера уведомлений

Класс `NotificationManager` предоставляет программный интерфейс для механизма уведомлений, встроенного в платформу Android. Уведомления отображаются в строке состояния в верхней части экрана устройства. Находясь в классе деятельности, можно создать экземпляр менеджера уведомлений следующим образом.

```
NotificationManager mgr =(NotificationManager)
              getSystemService(NOTIFICATION_SERVICE);
```

Эта строка кода создает объект `NotificationManager` путем вызова метода `getSystemService()`.

Создание уведомления

Для приложения Task Reminder необходим способ оповещения пользователя о том, что некоторая задача требует его внимания. Это происходит в момент, когда объект `AlarmManager` генерирует сигнал, ассоциированный с данной задачей. Отобразить сигнал в строке состояния можно с помощью менеджера уведомлений.

В методе `doReminderWork()` класса `ReminderService` введите код, показанный в листинге 14.1.

Листинг 14.1. Реализация метода `doReminderWork()`

```
Long rowId = intent.getExtras().
       getLong(RemindersDbAdapter.KEY_ROWID);           1

NotificationManager mgr = (NotificationManager)
        getSystemService(NOTIFICATION_SERVICE);         3

Intent notificationIntent = new Intent(this,
                ReminderEditActivity.class);            5
notificationIntent.putExtra(
             RemindersDbAdapter.KEY_ROWID, rowId);      6

PendingIntent pi = PendingIntent.getActivity(this, 0,
    notificationIntent, PendingIntent.FLAG_ONE_SHOT);   8

Notification note=new Notification(
        android.R.drawable.stat_sys_warning,
        getString(R.string.notify_new_task_message),
        System.currentTimeMillis());                    10

note.setLatestEventInfo(this, getString(
         R.string.notifiy_new_task_title), getString(
         R.string.notify_new_task_message), pi);        12

note.defaults |= Notification.DEFAULT_SOUND;            14
note.flags |= Notification.FLAG_AUTO_CANCEL;            15

// Если пользователь создаст более 2,147,483,647 задач,
// произойдет ошибка, потому что для типа int это
// максимальное значение. Вряд ли это когда-либо
// произойдет, но все же застрахуемся и от этого.
int id = (int)((long)rowId);                            19
mgr.notify(id, note);                                   20
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Намерение, запустившее службу `ReminderService`, содержит идентификатор текущей задачи. Он необходим для создания отложенного намерения. Когда уведомление выбрано в строке состояния, нужно, чтобы была запущена деятельность `ReminderEditActivity` с идентификатором, определяющим отложенное намерение. Тогда деятельность

`ReminderEditActivity` прочитает информацию о задаче и отобразит ее пользователю.

- ✓ **3.** Получение экземпляра `NotificationManager`.
- ✓ **5.** Создание нового намерения для класса `ReminderEditActivity`. Эта деятельность должна активизироваться, когда пользователь выбирает уведомление в строке состояния.
- ✓ **6.** Запись идентификатора задачи в намерение.
- ✓ **8.** Создание намерения, используемого системой уведомлений. Оно выполняется в другом процессе, поэтому необходимо отложенное намерение `PendingIntent`. Флажок `FLAG_ONE_SHOT` указывает на то, что отложенное намерение можно использовать только один раз.
- ✓ **10.** Создание уведомления, отображаемого в строке состояния. Конструктор класса уведомления получает следующие параметры.
  - **`icon: android.R.drawable.stat_sys_warning`.** Идентификатор ресурса значка, располагаемого в строке состояния. На значке изображен маленький треугольник с восклицательным знаком в центре. Это встроенный значок Android, поэтому нам не нужно беспокоиться о низком, среднем и высоком разрешениях, потому что платформа Android предоставляет все три варианта значка.
  - **`tickerText: getString(R.string.notify_new_task_message)`.** Текст, отображаемый при первой активизации уведомления.
  - **`when: System.currentTimeMillis()`.** Время, отображаемое в поле уведомления.
- ✓ **12.** Установка содержимого расширенного представления со стандартной компоновкой, предоставляемой операционной системой Android. Можно создать пользовательскую компоновку XML, но в данном примере я задал отображение обычного представления. Метод `setLatestEventInfo()` принимает следующие параметры.
  - **`context: this`.** Текущая деятельность служит контекстом, ассоциированным с информацией о событии.
  - **`contentTitle: getString(R.string.notify_new_task_title)`.** Заголовок расширенного представления.
  - **`contextText: getString(R.string.notify_new_task_message)`.** Текст, отображаемый в расширенном представлении.
  - **`contentIntent: pi`.** Намерение, запускаемое при выборе пользователем расширенного представления.
- ✓ **14.** Добавление звука в объект уведомления с помощью дизъюнктивной битовой маски. Воспроизводится звук, установленный для уведомлений по умолчанию. Конечно, звук слышен, только если в устройстве установлена достаточная громкость.
- ✓ **15.** Добавление флажка для удаления уведомления после его выбора пользователем. Используется аналогичная битовая маска.

- ✓ **19.** Приведение идентификатора `ID` (не путайте с идентификатором задачи) к целому числу. В базе данных SQLite идентификатор хранится как тип `long`, однако нужно привести его к типу `int`. Происходит потеря точности, однако вряд ли когда-либо будет установлено более 2 147 483 647 задач (это максимальное значение типа `int` в Java). Следовательно, приведение будет завершено успешно. Тип `int` необходим потому, что метод `notify` в строке 20 принимает только аргумент типа `int`.
- ✓ **20.** Активизация уведомления в строке состояния. Метод `notify()` принимает следующие параметры.
  - **`id: id`.** Идентификатор строки, уникальный на уровне приложения.
  - **`Notification: note`.** Объект уведомления, содержащий всю информацию о том, как выполняется уведомление пользователя.

Последовательность этапов уведомления

Процесс уведомления может зависеть от многих факторов, однако в общем случае он состоит из следующих этапов.

- ✓ Пользователь занят другим приложением, например просматривает электронную почту.
- ✓ Согласно расписанию, подходит время очередной задачи, хранящейся в Task Reminder, и операционная система генерирует сигнал об этом. Приложение создает уведомление в строке состояния.
- ✓ Пользователь может либо развернуть строку состояния и выбрать уведомление, либо проигнорировать его.
- ✓ Если пользователь разворачивает строку состояния и выбирает пункт уведомления, активизируется отложенное намерение, размещенное в уведомлении. Отложенное намерение запускает деятельность `ReminderEditActivity` с задачей, определяемой идентификатором строки.
- ✓ Уведомление удаляется из строки состояния.
- ✓ Информация о задаче извлекается из базы данных и отображается в форме деятельности `ReminderEditActivity`.

Добавление строковых ресурсов

Для нормальной работы уведомления необходимо добавить в файл строковых ресурсов `strings.xml` два новых ресурса.

- ✓ **`notify_new_task_message`.** Присвойте этому ресурсу значение `Просмотрите задачу!`. Эта строка отображается как сообщение в развернутом представлении строки состояния и применяется как текст сообщения при первом появлении уведомления.
- ✓ **`notify_new_task_title`.** Присвойте этому ресурсу значение `Task Reminder`, т.е. название приложения. Оно отображается как заголовок развернутого представления строки состояния.

Изменение уведомления

Иногда имеет смысл изменить представление уведомления в строке состояния. Рассмотрим следующую ситуацию. Предположим, выполняется некоторый фоновый код, проверяющий, отреагировал ли пользователь на уведомление. Код выявляет просроченные уведомления и, когда проходит два часа, отмечает их красным восклицательным знаком или серией вспышек светодиодного индикатора. К счастью, изменить уведомление подобным или другим образом несложно.

Если вызвать метод `notify()` с идентификатором, активным в строке состояния, и новым набором параметров уведомления, то уведомление будет обновлено в строке состояния. Следовательно, чтобы изменить уведомление, нужно лишь создать новый объект `Notification` со значком, содержащим красный восклицательный знак, и вызвать метод `notify()`, который обновит уведомление.

Удаление уведомлений

Пользователи — совершенно непредсказуемая публика. Ваш пользователь может находиться в любом месте земного шара и быть кем угодно — "чайником", опытным пользователем, программистом, ребенком, профессионалом или даже неграмотным человеком. В наше время нередки случаи, когда ребенок умеет пользоваться мобилкой или выходить в Интернет еще до того, как научится читать. Он знает лишь, где нужно щелкнуть, чтобы увидеть картинки или мультик. Каждый пользователь работает с устройством так, как ему заблагорассудится. Часто даже происходит следующее: пользователь видит уведомление, но не знает, что можно развернуть его непосредственно в строке состояния. Поэтому пользователь идет окольным путем: открывает панель запуска приложений, находит приложение Task Reminder, открывает его, находит задачу в списке задач и, наконец, открывает задачу, дабы узнать, что нужно сделать.

Если пользователь открывает приложение через панель запуска приложений, когда уведомление активное, оно остается активным. Даже если пользователь открывает задачу вручную, уведомление все равно остается в строке состояния, хоть это выглядит нелогично. Можно усовершенствовать приложение, добавив процедуру, которая распознает состояние задачи и удаляет ненужное уведомление из строки состояния. Однако если пользователь вручную открыл приложение и просматривает другую задачу, не представленную в строке состояния, процедура не должна удалять уведомление. Удалению подлежат только уведомления, которые пользователь просматривал после срабатывания их сигналов.

Менеджер уведомлений (класс `NotificationManager`) позволяет легко удалить существующее уведомление с помощью метода `cancel()`. Этот метод принимает единственный параметр — идентификатор уведомления. Как вы помните, в качестве идентификатора уведомления в Task Reminder используется идентификатор задачи в базе данных. Поэтому несложно открыть задачу и удалить уведомление, вызвав метод `cancel()` и передав ему идентификатор задачи.

Полезной может быть также возможность удалить все просроченные уведомления. Для этого достаточно вызвать метод `cancelAll()` класса `NotificationManager`.

Глава 15

Пользовательские настройки

В этой главе...

- Концепция настроек
- Отображение списка настроек
- Создание экрана настроек
- Класс PreferenceActivity
- Работа с настройками во время выполнения

Большинство программ можно сконфигурировать под потребности конкретного пользователя. Обычно я прилагаю немалые усилия на поиск установок, позволяющих сделать программу, с которой я часто работаю, более удобной и полезной. Разработчики программ предоставляют для этого много возможностей. Вы тоже, как разработчик приложений Android, должны предоставить пользователям возможность конфигурировать ваше приложение. К счастью, в Android встроены средства создания экранов, упрощающих редактирование настроек.

Платформа Android предлагает стабильную инфраструктуру настроек, позволяющую как декларативно, так и программно определять настройки разрабатываемого приложения. Операционная система постоянно хранит настройки в виде пар "ключ-значение" базовых типов данных. Вам не нужно хранить значения в файле, базе данных или любом другом месте. Инфраструктура настроек Android принимает значения и сохраняет их во внутреннем хранилище от имени приложения. В инфраструктуре настроек можно хранить булевы, вещественные, целочисленные и строковые значения. Настройки остаются неизменными в разных сеансах работы с приложением. Это означает, что после закрытия и повторного открытия приложения данные не пропадают. Настройки сохраняются даже в случае принудительного завершения процесса приложения.

В этой главе рассматривается инфраструктура настроек операционной системы Android и описываются способы ее использования в приложении. Для создания и редактирования настроек применяется встроенный класс `PreferenceActivity`. Я продемонстрирую процедуры чтения и записи настроек в коде приложения. В конце главы у нас будет полностью работоспособное приложение Task Reminder с интегрированными в него настройками.

Концепция настроек

Одна из наиболее приятных особенностей Android заключается в простоте разработки экрана, позволяющего модифицировать настройки. Значительная часть трудоемкой работы по кодированию выполняется операционной системой "за кулисами", без вашего участия. Вы должны всего лишь определить экран настроек в файле XML,

расположенном в папке res/xml текущего проекта. Этот файл отличается от файлов компоновки XML, расположенных в папке layout. Для определения настроек используется специальный формат, задающий структуру экранов, категорий и фактических настроек. В инфраструктуру встроены следующие общие настройки.

- ✓ **EditTextPreference.** Настройка, в которой можно хранить простой текст в строковом формате.
- ✓ **CheckBoxPreference.** Настройка, предназначенная для хранения булевого значения.
- ✓ **RingtonePreference.** Настройка, позволяющая хранить рингтон, доступный в устройстве.
- ✓ **ListPreference.** Настройка, позволяющая выбрать требуемый пункт в списке, приведенном в диалоговом окне.

Если указанных стандартных настроек недостаточно, можете создать собственную на основе базового класса Preference или DialogPreference. Класс DialogPreference предназначен для создания настроек на основе диалоговых окон. При щелчке на настройке открывается диалоговое окно с элементами управления, позволяющими менять значения настроек. Фактически стандартные настройки EditTextPreference и ListPreference являются диалоговыми окнами, наследующими класс DialogPreference.

Базовый класс PreferenceActivity позволяет загружать и редактировать настройки на экране деятельности так же, как на экране класса обычной деятельности Activity. Класс PreferenceActivity предоставляет события настроек и выполняет черновую работу, такую как установка элементов интерфейса типа EditTextPreference.

Отображение списка настроек

Базовый класс PreferenceActivity отображает иерархию объектов Preference в виде списков, возможно, на многих экранах (рис. 15.1).

Редактируемые настройки сохраняются с помощью экземпляра класса SharedPreferences. Этот класс предназначен для модификации данных настройки, возвращаемых методом getSharedPreferences() из любого объекта Context.

Класс PreferenceActivity очень похож на базовый класс обычной деятельности Activity, но работает немного иначе. Одна из наиболее важных особенностей класса PreferenceActivity — отображение настроек подобно системным настройкам. Таким образом достигается унификация внешнего вида и поведения всех пользовательских интерфейсов Android. Поэтому при работе с настройками в приложениях рекомендуется использовать класс PreferenceActivity, а не самодельные окна.

Хранение настроек

Android передает настройки в класс SharedPreferences, который автоматически сохраняет их во внутреннем хранилище. Поэтому создавать настройки несложно. Когда пользователь редактирует настройку, ее значение сохраняется автоматически, и программисту не нужно самому заботиться об этом.

На рис. 15.2 показано редактирование поля настройки `EditTextPreference`, используемой в приложении Task Reminder. После щелчка на кнопке OK Android передает введенное значение классу `SharedPreferences`, который сохраняет его. Программисту больше не нужно ничего делать, так как инфраструктура настроек сделала все остальное сама.

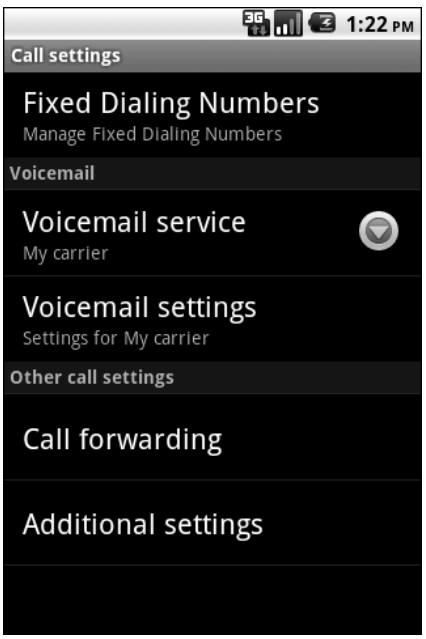


Рис. 15.1. Экран настроек приложения голосовой почты

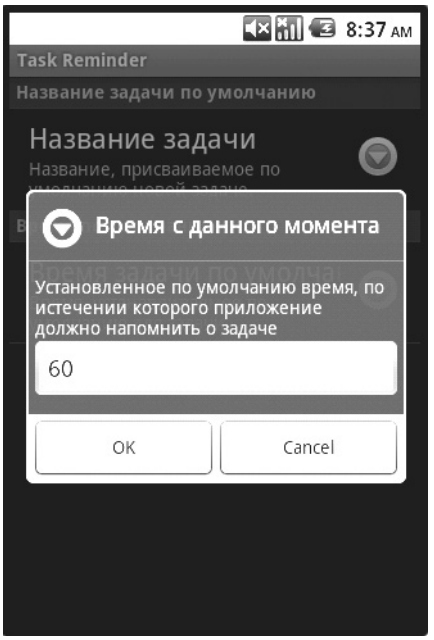


Рис. 15.2. Редактирование настроек приложения Task Reminder

Компоновка настроек

Иногда компоновка экранов Android может быть тяжелой, кропотливой работой, связанной с выравниванием, позиционированием, раскраской и другими операциями, находящимися больше в компетенции художника, чем программиста. Компоновка экрана Android — это почти то же самое, что верстка веб-страницы, на которой многие таблицы, используемые для позиционирования элементов, накладываются друг на друга. Скомпоновать экран иногда легко, а иногда очень тяжело. К счастью, компоновка настроек Android — намного более простая задача, чем в случае экранов приложений.

В Android экраны настроек разбиты на несколько категорий.

- ✓ **`PreferenceScreen`.** Этот класс представляет настройку верхнего уровня, являющуюся корнем иерархии настроек. Объекты `PreferenceScreen` можно использовать в двух местах.
  - *В деятельности* `PreferenceActivity`. В этом случае экран `PreferenceScreen` не отображается. Видны только настройки, размещенные в определении объекта `PreferenceScreen`.

- *В другой иерархии настроек*. В этом случае объект `PreferenceScreen` является шлюзом к другому экрану настроек. Можете представить себе это как структуру XML или как объявление объекта `PreferenceScreen`, вложенное в другое объявление `PreferenceScreen`. В документе XML элемент может быть вложен в другой элемент. Структурируя настройки, вы вкладываете элементы друг в друга с помощью объявлений объектов `PreferenceScreen`. Вкладывая их друг в друга, вы сообщаете операционной системе Android, в какой последовательности должны отображаться экраны настроек.

✓ **`PreferenceCategory`.** Используется для группирования объектов настроек и создания заголовков групп, описывающих каждую категорию.

✓ **`Preference`.** Настройка, отображаемая на экране. К этой категории относятся все стандартные настройки Android. Можете также создать собственную, т.е. пользовательскую настройку.

В файле компоновки XML можно комбинировать категории `PreferenceScreen`, `PreferenceCategory` и `Preference` любыми способами, вкладывая их друг в друга. Таким образом легко создать экран настроек, показанный на рис. 15.1.

Создание экрана настроек

Создание настроек с помощью класса `PreferenceActivity` и файла XML — довольно простая задача. В первую очередь нужно создать файл настроек XML, определяющий компоновку настроек, и строковые ресурсы, определяющие надписи на экране. Строковые ресурсы, представленные в элементах `TextView`, информируют пользователя о назначении настроек.

Создадим экран настроек `PreferenceScreen` для приложения Task Reminder. Предположим, пользователь должен иметь возможность установить с помощью настроек название задачи и время напоминания, предлагаемые по умолчанию для каждой новой задачи. В настоящий момент поле названия задачи пустое, а время напоминания установлено равным текущему системному времени устройства. Эти настройки позволят пользователю избежать нескольких дополнительных операций при создании новой задачи. Например, если установить время напоминания равным 60 минутам, то при создании задач, о которых нужно напомнить через 60 минут, пользователь может не открывать элемент установки времени. Ему ничего не нужно менять, так как в задаче автоматически будет сохранено время, установленное по умолчанию.

Создание файла настроек

Создайте в проекте папку `res/xml`, а в ней — файл `task_preferences.xml`. Содержимое файла приведено в листинге 15.1.

Листинг 15.1. Файл `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>
```

В листинге 15.1 введено довольно много строковых ресурсов. Их содержимое будет приведено в листинге 15.2, а пока что приведем краткое описание отмеченных строк кода XML.

- ✓ **2.** Корневой элемент экрана настроек. Служит контейнером для экрана устройства. В элементе `PreferenceScreen` размещены все объявления настроек.
- ✓ **4.** Определение категории параметров задач, устанавливаемых по умолчанию, например названия или описания. В строке 13 определена еще одна категория. Обычно такие настройки помещают в одну категорию, однако в данном случае я создал две категории, чтобы продемонстрировать, как можно использовать несколько элементов `PreferenceCategory` на одном экране.

- ✓ **5.** Определение ключа, используемого для сохранения и получения параметра из объекта `SharedPreferences`. Ключ должен быть уникальным.
- ✓ **6.** Определение категории для названия задачи.
- ✓ **7.** Определение элемента `EditTextPreference`, ответственного за сохранение настроек для названия задачи, предлагаемого по умолчанию.
- ✓ **8.** Ключ для названия задачи, предлагаемого по умолчанию.
- ✓ **9.** Класс `EditTextPreference` наследует базовый класс `DialogPreference`. Это означает, что при выборе настроек активизируется диалоговое окно настроек, аналогичное показанному на рис. 15.2. Эта строка кода определяет строку заголовка диалогового окна.
- ✓ **10.** Сообщение, выводимое в диалоговом окне настроек.
- ✓ **11.** Описание, выводимое на экране настроек (см. рис. 15.1).
- ✓ **12.** Название настройки.
- ✓ **13.** Определение класса категории `PreferenceCategory` для времени напоминания, установленного по умолчанию.
- ✓ **14.** Ключ категории.
- ✓ **15.** Название категории.
- ✓ **16.** Начало определения элемента `EditTextPreference`, который сохраняет время напоминания в минутах, устанавливаемое по умолчанию как интервал между текущим моментом и временем генерации сигнала.
- ✓ **17.** Ключ значения времени, установленного по умолчанию.
- ✓ **18.** Строка заголовка диалогового окна, отображаемого при выборе настройки.
- ✓ **19.** Сообщение, выводимое в диалоговом окне настроек.
- ✓ **20.** Описание настройки, отображаемое на главном экране настроек (см. рис. 15.1).
- ✓ **21.** Название настройки на экране настроек.

Добавление строковых ресурсов

Чтобы приложение успешно скомпилировалось, нужно добавить для настроек строковые ресурсы. В файле `res/values/strings.xml` добавьте следующие значения.

```
<!-- Preferences -->
<string name="pref_category_task_defaults_key">
        task_default_category</string>
<string name="pref_category_task_defaults_title">
        Название задачи по умолчанию</string>
<string name="pref_task_title_key">
        default_reminder_title</string>
<string name="pref_task_title_dialog_title">
```

```
        Название задачи</string>
<string name="pref_task_title_message">
        Название, присваиваемое задаче по умолчанию.
        </string>
<string name="pref_task_title_summary">
       Название, присваиваемое по умолчанию новой задаче.
        </string>
<string name="pref_task_title_title">
        Название задачи</string>
     -- это стиль КОД
<string name="pref_category_datetime_key">
        date_time_default_category</string>
<string name="pref_category_datetime_title">
        Время по умолчанию</string>
<string name="pref_default_time_from_now_key">
        time_from_now_default</string>
<string name="pref_default_time_from_now_dialog_title">
        Время с данного момента</string>
<string name="pref_default_time_from_now_message">
        Установленное по умолчанию время в минутах,
        по истечении которого приложение должно напомнить
        о задаче</string>
<string name="pref_default_time_from_now_summary">
        Время, устанавливаемое по умолчанию для задачи.
        </string>
<string name="pref_default_time_from_now_title">
        Время задачи по умолчанию</string>
```

Теперь приложение должно скомпилироваться.

Таким образом, для определения экрана настроек необходимо всего лишь задать значения атрибутов. Экран настроек можно определить в файле XML, но это еще не означает, что настройки появятся на экране устройства. Для этого нужно создать класс на основе базового класса `PreferenceActivity`.

Класс PreferenceActivity

Этот класс отображает иерархию настроек на экране устройства соответственно определению настроек в файле XML, который вы только что создали. Настройки могут размещаться на многих экранах (если создано много вложенных объектов `PreferenceScreen`). Все настройки автоматически сохраняются классом `SharedPreferences`. Автоматически отображаемые настройки выводятся с применением того же стиля, что и системные настройки. Благодаря этому легко унифицировать их внешний вид, чтобы пользователь чувствовал себя комфортно при работе с настройками.

Чтобы отобразить только что созданный вами экран `PreferenceScreen`, добавьте в приложение деятельность, производную от базового класса `PreferenceActivity`. Присвойте ей имя `TaskPreferences`. Код деятельности `TaskPreferences` показан в листинге 15.2.

Листинг 15.2. Файл `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));           6
  timeDefault.getEditText().
     setKeyListener(DigitsKeyListener.getInstance());           7
  }
}
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Заголовок определения класса `TaskPreferences`, наследующего базовый класс `PreferenceActivity`.
- ✓ **5.** Вызов метода добавления настроек из ресурсов. Метод получает идентификатор ресурса в файле `task_preferences.xml`.
- ✓ **6.** Извлечение настройки с помощью ключа, определенного в файле `task_preferences.xml`.
- ✓ **7.** Получение объекта `EditText`, наследующего класс `EditTextPreference`, с помощью метода `getEditText()`. Через объект `EditText` устанавливается приемник событий клавиатуры, ожидающий нажатия клавиш. Приемник клавиатуры устанавливается с помощью метода `setKeyListener()`, получающего экземпляр `DigitsKeyListener`, поскольку объект `EditTextPreference` в данном случае должен позволять вводить только цифры. Использование объекта `DigitsKeyListener` гарантирует, что в настройки будут записаны только цифры.

Деятельность `TaskPreferences` готова к использованию. Базовый класс `PreferenceActivity` позволяет пользователям редактировать и сохранять настройки. Как видите, реализация настроек потребовала написания очень малого объема кода. Следующий этап — добавление меню, с помощью которого можно открыть экран настроек.

Объявите вашу новую деятельность `TaskPreferences` в манифесте приложения `AndroidManifest.xml`, добавив следующую строку.

```
<activity android:name=".TaskPreferences"
                android:label="@string/app_name" />
```

Активизация класса PreferenceActivity

Чтобы активизировать деятельность настроек, нужно добавить пункт меню в деятельность ReminderListActivity. Для этого необходимо добавить новое определение меню в файл list_menu.xml, расположенный в папке res/menu. При обновлении этого файла будет обновлено меню деятельности ReminderListActivity. Ниже показан код обновленного файла 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" />
    <item android:id="@+id/menu_settings"
      android:icon=
                "@android:drawable/ic_menu_preferences"
      android:title="@string/menu_settings" />
</menu>
```

Элемент, отмеченный полужирным шрифтом, добавляет в меню пункт открытия экрана настроек. В этом пункте используется встроенный значок Android и строковый ресурс menu_settings. Этот строковый ресурс нужно добавить в файл strings.xml и присвоить ему значение Настройки.

Обработка выбора пункта меню

Итак, меню обновлено. Теперь необходимо запрограммировать реакцию на выбор пункта меню. Для этого добавьте в метод unMenuItemSelected() деятельности ReminderListActivity код, отмеченный полужирным шрифтом.

```
@Override
public boolean onMenuItemSelected(int featureId,
                                  MenuItem item) {
  switch(item.getItemId()) {
    case R.id.menu_insert:
      createReminder();
      return true;
    case R.id.menu_settings:
      Intent i = new Intent(this, TaskPreferences.class);
      startActivity(i);
      return true;
    }
  return super.onMenuItemSelected(featureId, item);
}
```

Инструкции, отмеченные полужирным шрифтом, создают объект намерения Intent с целевым классом TaskPreferences. Когда пользователь выбирает в меню пункт Настройки, активизируется экран настроек, с помощью которого пользователь может редактировать их. Если запустить приложение Task Reminder, открыть меню и выбрать пункт Настройки, откроется экран, показанный на рис. 15.3.

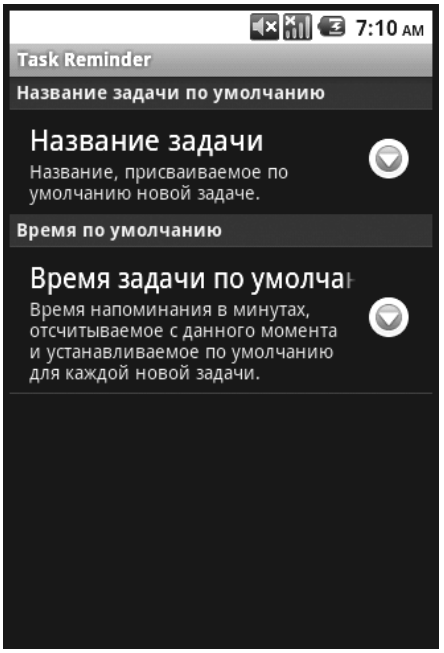


Рис. 15.3. Экран настроек

Работа с настройками во время выполнения

Установка настроек в объекте `PreferenceActivity` — важный этап, но, чтобы они были полезными для пользователя, нужно предоставить ему возможность читать настройки из объекта `SharedPreferences` во время выполнения. Нужно также каким-либо образом применить настройки в приложении. К счастью, платформа Android существенно упрощает обе эти задачи.

В приложении Task Reminder необходимо прочитать значения настроек в объекте `ReminderEditActivity` и установить их в задачу, когда пользователь начинает создавать ее. Настройки хранятся в объекте `SharedPreferences`, поэтому к ним можно обратиться в любой деятельности приложения.

Извлечение настроек

Откройте в окне редактора файл деятельности `ReminderEditActivity` и найдите метод `populateFields()`. Этот метод выясняет, какая задача открыта для редактирования — существующая или новая (создаваемая). Если задача новая, нужно извлечь значения, установленные по умолчанию, из объекта `SharedPreferences` и загрузить их в деятельность. Если пользователь не устанавливал настройки, они будут представлены пустыми строками. В данный момент проигнорируем их. Будем использовать настройки, только если пользователь устанавливал их.

Извлечем значения настроек из объекта `SharedPreferences` (листинг 15.3). В методе `populateFields()` добавьте код, отмеченный полужирным шрифтом.

Листинг 15.3. Извлечение значений настроек из объекта `SharedPreferences`

```
private void populateFields() {
  if (mRowId != null) {
    Cursor reminder = mDbHelper.fetchReminder(mRowId);
    startManagingCursor(reminder);
    mTitleText.setText(reminder.getString(
    reminder.getColumnIndexOrThrow(
                    RemindersDbAdapter.KEY_TITLE)));
    mBodyText.setText(reminder.getString(
    reminder.getColumnIndexOrThrow(
                    RemindersDbAdapter.KEY_BODY)));
    SimpleDateFormat dateTimeFormat = new
      SimpleDateFormat(DATE_TIME_FORMAT);
    Date date = null;
    try {
      String dateString =
       reminder.getString(reminder.getColumnIndexOrThrow(
         RemindersDbAdapter.KEY_DATE_TIME));
      date = dateTimeFormat.parse(dateString);
      mCalendar.setTime(date);
    } catch (IllegalArgumentException e) {
      e.printStackTrace();
    } catch (ParseException e) {
      e.printStackTrace();
    }
  } else {                                                    21
  SharedPreferences prefs =
    PreferenceManager.
              getDefaultSharedPreferences(this);              22
    String defaultTitleKey =
          getString(R.string.pref_task_title_key);            23
    String defaultTimeKey = getString(
        R.string.pref_default_time_from_now_key);             24
    String defaultTitle =
            prefs.getString(defaultTitleKey, "");             26
    String defaultTime =
             prefs.getString(defaultTimeKey, "");             27
    if("".equals(defaultTitle) == false)
      mTitleText.setText(defaultTitle);                       30
    if("".equals(defaultTime) == false)
      mCalendar.add(Calendar.MINUTE,
                    Integer.parseInt(defaultTime));           33
  }
  updateDateButtonText();
  updateTimeButtonText();                                     37
}
```

Ниже приведено описание отмеченных новых строк кода.

- ✓ **21.** В блоке `else` находятся инструкции, реализующие новую задачу.
- ✓ **22.** Извлечение объекта `SharedPreferences` с помощью метода `getDafaultSharedPreferences` класса `PreferenceManager`.
- ✓ **23.** Извлечение из строкового ресурса ключа для установленного по умолчанию названия задачи. Этот же ключ используется в листинге 15.1 для определения настроек.
- ✓ **24.** Извлечение ключа для смещения времени, извлекаемого из настройки (ключ другой, но процесс тот же, что и в строке 23).
- ✓ **26.** Извлечение установленного по умолчанию названия задачи из настроек путем вызова метода `getString()` объекта `SharedPreferences`. Первый параметр — ключ настройки, а второй — значение, установленное по умолчанию. Если настройка не существует или не устанавливалась, возвращается пустая строка.
- ✓ **27.** Извлечение из настройки времени с помощью того же метода, что и в строке 26, но с другим ключом.
- ✓ **30.** Установка текстового значения в представлении `EditText`, содержащем название задачи. Эта операция выполняется, только если значение настройки не равно пустой строке.
- ✓ **33.** Увеличение значения локального объекта `Calendar` путем вызова метода `add()` с параметром `Calendar.MINUTES` (если значение настройки не равно пустой строке). Константа `Calendar.MINUTES` информирует объект `Calendar` о том, что второй параметр приведен в минутах. Если при добавлении указанного количества минут изменяются часы, дни и другие поля календаря, объект `Calendar` обновляет их. Например, если объект `Calendar` содержал значение `2010-12-31 11:45 pm` и метод `add()` добавил 60 минут, новое значение будет равно `2011-01-01 12:45 pm`. В объекте `EditTextPreference` все значения хранятся в строковом формате, поэтому количество минут преобразуется в целочисленный формат с помощью метода `Integer.parseInt()`. При изменении времени в локальном объекте `Calendar` должны быть обновлены надписи на кнопках выбора даты и времени.
- ✓ **37.** Инструкции, обновляющие надписи на кнопках выбора даты и времени для синхронизации надписей с содержимым объекта `Calendar`.

Сейчас при запуске приложения можно в эмуляторе установить настройки и увидеть их при создании новой задачи на экране деятельности `ReminderEditActivity`. Попробуйте очистить настройки (присвоив им пустые строки), и вы увидите, что для новой задачи значения по умолчанию не применяются.

Программная установка настроек

В приложении Task Reminder установка настроек в коде не применяется, но мы рассмотрим этот прием ввиду его важности на практике. Рассмотрим следующую ситуацию. Предположим, вы разрабатываете для регистрационной службы компании приложение, где пользователь должен ввести название отдела, в котором он работает. В приложении есть настройка, содержащая название отдела, предлагаемое по умолчанию. Однако пользователь не открывает экран настроек, потому что это не его приложение. Он лишь заполняет форму. Если регистрационная служба, согласно плану ее деятельности, обслуживает отделы по очереди, то можно запрограммировать вставку в настройки многих параметров предыдущего пользователя (в частности, название отдела), чтобы следующему служащему из того же отдела не нужно было вводить параметры вручную. Тогда пользователю нужно будет лишь подтвердить предлагаемое значение.

Чтобы редактировать настройки программно, нужен экземпляр класса `SharedPreferences`. Его можно получить из объекта `PreferenceManager`, как показано в листинге 15.4. Для редактирования настроек в объекте `SharedPreferences` из него нужно извлечь объект `Editor`. После редактирования настроек их нужно зафиксировать, что тоже продемонстрировано в листинге 15.4.

Листинг 15.4. Программное редактирование настроек

```
SharedPreferences prefs =
  PreferenceManager.getDefaultSharedPreferences(this);  1
Editor editor = prefs.edit();                            2
editor.putString("default_department", "Отдел КБ-4");    3
editor.commit();                                         4
```

Ниже приведено описание отмеченных строк кода.

- ✓ **1.** Извлечение экземпляра `SharedPreferences` из объекта `PreferenceManager`.
- ✓ **2.** Извлечение объекта `Editor` путем вызова метода `edit()` объекта `SharedPreferences`.
- ✓ **3.** Редактирование настройки, имеющей ключ `default_department`, с помощью метода `putString()` объекта `Editor`. Настройке присваивается значение `Отдел КБ-4`. Обычно ключевое значение извлекается из строкового ресурса, а новое значение настройки — из другой части программы, однако здесь для простоты они представлены строковыми литералами.
- ✓ **4.** Фиксация изменений, внесенных в настройку, с помощью метода `commit()` объекта `Editor`. После фиксации новое значение сохраняется в объекте `SharedPreferences`. Операция фиксации автоматически заменяет текущее значение, хранящееся с указанным ключом в объекте `SharedPreferences`, новым значением, заданным при вызове метода `putString()`.

Если не вызвать метод `commit()` объекта `Editor`, изменения настроек не будут сохранены, и приложение будет работать не так, как ожидается.

Добавив экран настроек в приложение, вы сделали его конфигурируемым и, следовательно, более полезным для пользователей. Добавить новые настройки несложно в коде или файле объявлений XML, поэтому не упускайте возможность сделать приложение более мощным и удобным.

Часть IV

Великолепные десятки

The 5th Wave — Рич Теннант

В этой части...

Часть IV содержит тщательно охраняемые "секреты" платформы Android, о которых можно узнать только в результате многолетнего опыта работы с этой операционной системой. В первую очередь я приведу список демонстрационных приложений, которые могут служить образцами для начинающих разработчиков. В них используются многие средства Android, начиная с баз данных SQLite и заканчивая программными интерфейсами веб-приложений, разработанными сторонними производителями. В конце данной части вы найдете список профессиональных инструментов и библиотек, существенно облегчающих создание приложений Android. С их помощью можно создавать более мощные приложения и повысить производительность труда.

Глава 16

Десять бесплатных приложений и средств разработки

На протяжении всей карьеры профессионального разработчика приложений Android перед вами постоянно будут возникать преграды, которые невозможно преодолеть самостоятельно. Не помогут ни мастерство, ни прошлый опыт, ни изобретательность. Единственный выход — посмотреть, как это делают другие люди. Часто вам будет казаться, что вы нашли правильное решение проблемы, хотя существует более простое и эффективное решение, которое можно увидеть в образцах кода, созданных другими людьми. Например, как запрограммировать обнаружение столкновений в играх или взаимодействие с объектами JSON? Сталкиваясь с подобными проблемами, я обычно немедленно отправляюсь в Интернет искать образцы кодов. Вероятность того, что кто-то уже сталкивался с этой проблемой, всегда очень высокая.

Образцы кодов — прекрасная штука! Они, конечно, не готовы к немедленному использованию в вашем приложении, но в них можно найти нужное решение. Более того, они служат великолепным обучающим пособием по программированию приложений для Android. Например, как программно запретить редактирование ячеек таблицы, созданной объектом `Table`? Есть метод `isEditable()`, но нет ни одного метода или свойства, позволяющего запретить редактирование. Если вы попытаетесь самостоятельно решить эту проблему, читая учебники и описание класса `Table`, то, скорее всего, нагромоздите десятки строк кода, извлекающих свойства столбцов и ячеек. Между тем, достаточно добавить в оператор создания таблицы `new` одну короткую строку, переопределяющую метод `isEditable()`. Включите в тело метода единственный оператор `return false`, и все ячейки станут недоступными для редактирования.

Много образцов кодов есть в пакете Android SDK, поставляемом в составе дистрибутива Android. В главе 2 упоминались образцы использования библиотек, приведенные в SDK, однако настоящий кладезь знаний — коды реальных приложений, которые можно найти в Интернете. Там вы найдете примеры на все случаи жизни и решение любых проблем благодаря открытости платформы Android.

Однако всего лишь сказать: “Попробуйте поискать” было бы мало, не так ли? Чтобы ускорить процесс вашего обучения, в данной главе представлены десять приложений с открытыми кодами, которые могут служить великолепными образцами при разработке приложений. Большинство из них — реальные приложения, которые можно загрузить из Android Market и установить в своем устройстве. Рекомендую сначала установить приложение и посмотреть, как оно работает, тогда вам будет легче понять его код. Не забывайте также о Google (или другом поисковом механизме, например Yandex). Введите в поле поиска `Android` и еще несколько ключевых слов, характеризующих проблему, и вы получите список из сотен сайтов, на которых упоминается данная проблема.

Foursquare

В данный момент наблюдается настоящий бум популярности этого приложения социальных сетей[1]. Оно позволяет пользователям отмечаться на местности и на основе GPS отслеживать текущее местоположение своих друзей во всем мире. При соблюдении определенных условий участнику сети может быть присвоено звание мэра определенного места (кафе, парка, дискотеки и т.п.). Участники награждаются значками разных типов. При активном участии в сети можно получить статус суперпользователя, предоставляющий многие права по конфигурированию правил игры и местности. Вам как разработчику приложений Android будет интересно ознакомиться с исходными кодами данного приложения на сайте Google Code. Вы узнаете, как выполняется взаимодействие с интерфейсами сторонних производителей, возвращающими документы XML и JSON. В этом источнике вы найдете огромный ресурс великолепных решений и примеров использования средств Android, включая следующие:

- ✓ асинхронные задачи;
- ✓ синтаксический разбор документов XML;
- ✓ использование нескольких деятельностей;
- ✓ аутентификация пользователей средствами OAuth;
- ✓ инструменты Google Maps и Map Layers;
- ✓ GPS;
- ✓ интеграция с веб-интерфейсами сторонних производителей.

Вы найдете не просто огромное количество исходных кодов, но и хорошо организованную структуру приложения, иерархию пакетов, облегчающую поиск нужных примеров. Откройте страницу по адресу `http://code.google.com/p/foursquared`.

LOLCat

Великолепный пример приложения Android, манипулирующего изображениями. Изучая его исходный код, вы узнаете, как программно получить доступ к камере устройства, добавить заголовок, сохранить изображение на карте SD и пр. Вы увидите, как создаются намерения, передающие изображения через MMS и вложения электронных писем. Адрес сайта — `http://code.google.com/p/apps-for-android`.

Amazed

Это компьютерная игра, в которой можно наблюдать использование встроенного акселерометра для управления двухмерными шариками в лабиринте постоянно усложняемых препятствий. Если вам необходимо изучить способы эффективного взаимодействия с акселерометром, то очень поможет исходный код этой игры. В нем также реализованы алгоритмы обнаружения столкновений и ветвления ситуаций игры. Исходный код игры находится по адресу `http://code.google.com/p/apps-for-android`.

[1] В сентябре 2011 года выпущена русская версия Foursquare. — *Примеч. ред.*

Примеры использования API-функций

Пакет Android SDK содержит множество примеров приложений, в частности, примеров, демонстрирующих использование API-функций. Большинство примеров небольшие и легкие для понимания. Образцы их исходных кодов можно использовать в разрабатываемых приложениях. Если у вас есть много идей, но мало времени, воспользуйтесь образцами кодов. Я рекомендую установить демонстрационные приложения в устройство и попробовать поработать с ними. Демонстрационные примеры находятся в папке `samples` пакета Android SDK.

Пример MultiResolution

Если вам нужно разработать приложение, хорошо работающее на экранах разных размеров, ознакомьтесь с примером `MultiResolution`, приведенным в Android SDK. Он позволит вам сэкономить много часов работы, потраченных на позиционирование и отладку представлений пользовательского интерфейса. Пакет Android SDK содержит работоспособный пример, демонстрирующий поддержку множества разрешений и размеров экрана. В примере показаны правильные приемы учета размеров в ресурсах и гибкого позиционирования элементов интерфейса без разного рода уловок, загромождающих код. Настоятельно рекомендую ознакомиться с этим примером до того, как начнете разрабатывать свое первое реальное приложение. Правильный подход к вопросу разрешения экрана избавит вас от множества проблем и поможет сэкономить рабочее время. Пример находится в папке `samples/MultiResolution`.

Пакет Last.fm

Если вы разрабатываете приложения, в которых используются потоковые аудиосредства, вам обязательно нужно ознакомиться с библиотекой Last.fm API. Для запуска и тестирования демонстрационных приложений нужно приобрести ключ Last.fm API, зарегистрировавшись на сайте `http://last.fm/api/account`. Вам понадобится также платная учетная запись на сайте потоковой музыки. Однако для просмотра исходных кодов платная учетная запись не нужна. Если вы хотите всего лишь получить исходные коды, вам не понадобится даже ключ. Данный пример поможет понять принципы обработки потоковых музыкальных ресурсов, находящихся на удаленном сайте. Образцы кодов можно скачать с сайта `http://github.com/mxcl/lastfm-android`.

Hubroid

Git — это популярная открытая DVCS (Distributed Version Control System — распределенная система управления версиями). Фактически все коды и документы, использованные в данной книге, в процессе ее написания находились в разных хранилищах Git. Приложение Hubroid работает на базе GitHub.com и предназначено для просмотра хранилищ Git с помощью портативного устройства. В приложении Hubroid демонстрируется использование GitHub API. Просматривая его исходные коды, вы поймете, как работает Git. Исходные коды доступны по адресу `http://github.com/eddieringle/hubroid`.

Facebook SDK для Android

Если вы достаточно амбициозны, то можете затеять такое грандиозное предприятие, как создание приложения для Facebook, но, возможно, вы не знаете, с чего начать. Впрочем, можете взяться за работу и полегче. Пакет инструментов Facebook Android SDK позволяет легко интегрировать функциональность Facebook в любое приложение Android. Можете использовать его для авторизации посетителей, создания запросов, сохранения записей и решения многих других задач. Посетите сайт `http://github.com/facebook/facebook-android-sdk`.

Replica Island

Возможно, вы хотите создать игру с боковой прокруткой, но не знаете с чего начать. Сейчас это уже не проблема, потому что можно загрузить исходные коды Replica Island — крутой игры с нашим любимым персонажем — зеленым механическим человечком по имени Андроид. Эта популярная бесплатная игра представлена на сайте Android Market. Ее исходный код (доступен на странице `http://code.google.com/p/replicaisland`) полностью открытый и может служить прекрасным примером для разработчиков.

Учебник по SQLite

Если вы хотите быстро освоить основы баз данных SQLite без сложных технологий типа служб, фоновых задач и т.п., ознакомьтесь с приложением Notepad, разработанным для платформы Android. Оно довольно простое, тем не менее его исходный код поможет вам понять принципы работы SQLite. Посетите страницу `http://d.android.com/guide/tutorials/notepad/index.html`.

Глава 17

Десять инструментов, которые облегчат вашу жизнь

Как разработчик, вы часто будете создавать для себя вспомогательные инструменты, чтобы сделать свой труд более производительным. Например, я создал огромное количество вспомогательных методов и классов для работы с асинхронными соединениями, документами XML и JSON, датами и временем и т.п. Однако, прежде чем самостоятельно заниматься разработкой, загляните в Интернет. Возможно, нужные вам инструменты уже существуют. Причем они, скорее всего, окажутся лучшими, чем ваши "самодельные" классы, потому что их создавали профессиональные разработчики, которые затратили на эту работу много времени и предусмотрели много различных ситуаций. В данной главе я описал десять инструментов и утилит, которые облегчат вашу жизнь и повысят производительность вашего труда и качество разрабатываемых вами приложений.

droid-fu

Это открытая библиотека классов, которые помогут радикально уменьшить время разработки. Библиотека содержит вспомогательные классы, выполняющих вместо вас всю рутинную работу, такую как фоновая обработка асинхронных запросов, загрузка изображений из Интернета и, как это ни удивительно, управление жизненным циклом приложения. Работая с `droid-fu`, можете не беспокоиться об изменяющихся состояниях приложения, потому что библиотека возьмет на себя их отслеживание. Можно не только применять библиотеку `droid-fu`, но и модифицировать ее классы, потому что ее исходные коды открыты и приведены на сайте `http://github.com/kaeppler/droid.fu`.

RoboGuice

Нет, это не источник энергетических соков для роботов. Это инфраструктура библиотек компании Google, облегчающих управление зависимостями. Представленные в библиотеках средства инициализируют переменные в нужные моменты времени. Их использование позволит существенно уменьшить объем кода и облегчить поддержку приложения в процессе эксплуатации. Посетите сайт `http://code.google.com/p/roboguice`.

DroidDraw

Это инструмент создания графических пользовательских интерфейсов Android. Компоновка элементов интерфейса выполняется путем их перетаскивания с панели

компонентов. Позволяет визуализировать пользовательский интерфейс, не компилируя приложение. Интерфейс, созданный в DroidDraw, можно сохранить и перенести в рабочую среду Eclipse. На момент написания данной книги была доступна только бета-версия DroidDraw. Сайт программы — `www.droiddraw.org`.

Draw 9-patch

Утилита, позволяющая создавать масштабируемые изображения для Android. В данной книге изображения Draw 9-patch не рассматривались. Информацию о них можно найти по такому адресу:

```
http://d.android.com/guide/developing/tools/draw9patch.html
```

С помощью утилиты Draw 9-patch можно встраивать в изображение инструкции, сообщающие операционной системе, как и когда можно растягивать изображение, чтобы оно было отображено с высоким качеством при любых разрешениях экрана.

Hierarchy Viewer

При работе с представлениями разного типа в файле компоновки часто бывает тяжело понять, как они будут выглядеть на экране. Утилита Hierarchy Viewer, находящаяся в папке `tools` пакета Android SDK, позволяет увидеть точное графическое представление виджетов на экране. Хорошо видны границы виджетов, что позволяет ясно представить, как они расположены по отношению друг к другу. С помощью утилиты можно настраивать интерфейс с точностью до пикселей. Эта утилита позволяет увеличивать изображение интерфейса, дабы убедиться в том, что он прорисовывается без дефектов на экранах с разными разрешающими способностями. Дополнительную информацию об этой утилите можно найти по такому адресу:

```
http://developer.android.com/guide/developing/tools/
hierarchy-viewer.html
```

Application Exerciser Monkey

Эта утилита используется для стресс-тестирования приложений Android. Она имитирует случайные щелчки, прикосновения и другие пользовательские события, дабы убедиться в том, что неправильное использование приложения не приводит к его краху. Тестировать приложение можно как в эмуляторе, так и на реальном устройстве. Дополнительная информация приведена по такому адресу:

```
http://developer.android.com/guide/developing/tools/monkey.html
```

zipalign

Утилита для выравнивания несжатых данных в файле APK. Минимизирует потребление памяти во время выполнения. Если в Eclipse используется надстройка ADT, приложение всегда выравнивается при экспорте в формат APK, как было показано

в главе 8. Следовательно, утилита zipalign в этом случае не нужна. Дополнительную информацию можно найти по такому адресу:

```
http://developer.android.com/guide/developing/tools/zipalign.html
```

layoutopt

Утилита командной строки, анализирующая компоновку приложения и сообщающая о проблемах и неэффективных процедурах. Этот инструмент проходит по всем папкам компоновок и ресурсов и обнаруживает недостатки, которые могут замедлить выполнение приложения. Откройте следующую страницу:

```
http://developer.android.com/guide/developing/tools/layoutopt.html
```

Git

Бесплатная распределенная система управления версиями, представляющая собой сверхбыстрый инструмент манипулирования хранилищами версий. Существенно облегчает ветвление версий. Ветви легко интегрируются в главный поток версий. В Eclipse существуют надстройки, позволяющие управлять хранилищами Git непосредственно в рабочей среде Eclipse. Поскольку Git является распределенной системой, хранилища могут находиться на удаленных серверах. Можете бесплатно установить свои закрытые хранилища на сайтах Projectlocker.com и Unfuddle.com. Если ваши коды открытые, можете создать бесплатное хранилище на сайте Github.com. Дополнительная информация приведена на сайте `www.git-scm.com`.

Paint.NET и GIMP

В процессе разработки приложений для Android вам придется часто работать с изображениями. Большинство профессиональных дизайнеров используют для этого программу Adobe Photoshop, но она довольно дорогая. Кроме того, большинство ее возможностей не нужны для приложений, работающих в мобильных устройствах. К тому же, есть две хорошие бесплатные графические программы — Paint.NET и GIMP.

Paint.NET — это бесплатный графический редактор на основе .NET Framework, широко применяемый разработчиками всего мира. Его можно использовать в Windows для создания любых рисунков, в том числе для платформы Android. Загрузить Paint.NET можно с сайта `www.getpaint.net`.

GIMP — это открытый бесплатный аналог программы Photoshop. Устанавливается в Windows, Linux и Mac. Сайт программы — `www.gimp.org`.

Предметный указатель

Л

М

Н

О

П

Р

С

Т

У

Х

Android 3 для профессионалов
Создание приложений для планшетных компьютеров и смартфонов

Сатия Коматинени, Дэйв Маклин, Саид Хашими

www.williamspublishing.com

Эта книга посвящена построению реальных мобильных приложений с использованием новой версии Android 3.0 SDK. В ней раскрываются все аспекты, начиная с основ создания приложений для встроенных устройств, телефонов и планшетных ПК и заканчивая расширенными концепциями, такими как построение трехмерных компонентов и реализация многозадачности. Кроме того, вы научитесь работать с картами, реализовать средства поиска, использовать виджеты домашнего экрана и ActionBar.

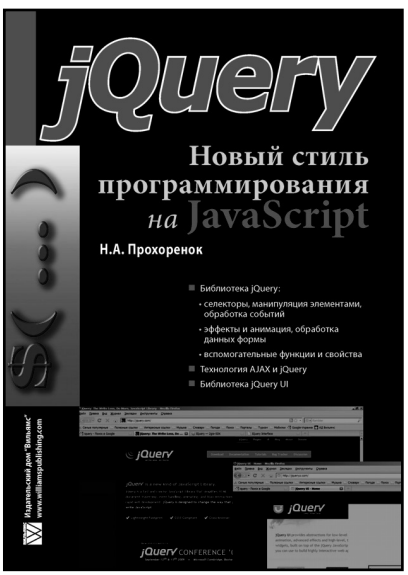
Благодаря этому руководству и многочисленным советам экспертов, вы быстро научитесь создавать современные мобильные приложения и запускать их на десятках основанных на Android смартфонов. Вы освоите множество API-интерфейсов Android, включая интерфейсы, которые предназначены для работы с медиа и датчиками. Вы ознакомитесь с новыми возможностями Android 3.0, в числе которых улучшенный пользовательский интерфейс для всех платформ Android, интеграция с WebM и многое другое, и обретете знания, которые помогут строить великолепные передовые приложения и быстро реагировать на изменения в будущем.

ISBN 978-5-8459-1746-1

в продаже

JQUERY
Новый стиль программирования на JavaScript

Н.А. Прохоренок



www.williamspublishing.com

Книга является справочником по JavaScript-библиотеке jQuery. Рассматриваются функциональные возможности библиотеки, полезные для максимально широкого круга задач, включая механизм селекторов, манипулирование параметрами и содержимым элементов DOM-модели документа, обработку событий и данных форм. Продемонстрированы возможности использования технологии AJAX для обмена данными с сервером без перезагрузки страницы. Описаны как базовые свойства и методы объекта XMLHttpRequest, так и интерфейс доступа к AJAX, предоставляемый библиотекой jQuery. Кроме того в книге рассматривается библиотека визуальных компонентов jQuery UI, предоставляющая готовые решения, которые может использовать любой разработчик, даже не владея основами jQuery и JavaScript. Эта библиотека позволяет создавать в документе нестандартные компоненты, панели с вкладками и др.

ISBN 978-5-8459-1603-7

в продаже

ПОЛНЫЙ СПРАВОЧНИК ПО JAVA™ 7-Е ИЗДАНИЕ

Герберт Шилдт

www.williamspublishing.com

Книга известного “гуру” в области программирования посвящена новой версии одного из наиболее популярных и совершенных языков – Java. Построенная в виде учебного и справочного пособия, она является превосходным источником исчерпывающей информации по последней версии платформы Java, Java SE 6, и позволяет практически с нуля научиться разрабатывать приложения и аплеты производственного качества. Помимо синтаксиса самого языка и фундаментальных принципов программирования, в книге подробно рассматриваются такие сложные вопросы, как ключевые библиотеки Java API, каркас коллекций, создание аплетов и сервлетов, AWT, Swing и Java Beans. Немалое внимание уделяется вводу-выводу, работе в сети, регулярным выражениям и обработке строк.

ISBN 978-5-8459-1168-1

в продаже

JAVA™ 2
БИБЛИОТЕКА ПРОФЕССИОНАЛА
ТОМ 2. ТОНКОСТИ ПРОГРАММИРОВАНИЯ
ВОСЬМОЕ ИЗДАНИЕ

Кей Хорстманн
Гари Корнелл

www.williamspublishing.com

Книга ведущих специалистов по программированию на языке Java представляет собой обновленное издание фундаментального труда, учитывающее всю специфику новой версии платформы Java SE 6. Подробно рассматриваются такие темы, как новые средства ввода-вывода, использование XML, API-интерфейсы работы в сети и взаимодействия с базами данных (JDBC), интернационализация, построение графических интерфейсов, безопасность, JavaBeans, взаимодействие с распределенными объектами и кодом, написанным на языке сценариев, а также платформенно-ориентированные методы, позволяющие обращаться на низком уровне к функциям операционной системы. Книга изобилует множеством примеров, которые не только иллюстрируют концепции, но также демонстрируют способы правильной разработки, применяемые в реальных условиях.

ISBN 978-5-8459-1482-8 **в продаже**

JAVA™ 2
БИБЛИОТЕКА ПРОФЕССИОНАЛА
ТОМ 1. ОСНОВЫ
8-Е ИЗДАНИЕ

Кей Хорстманн
Гари Корнелл

www.williamspublishing.com

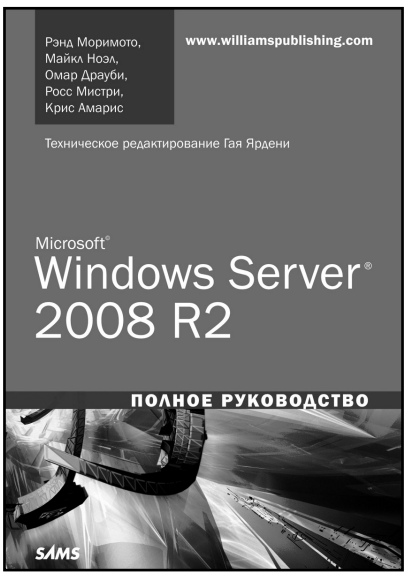
Книга ведущих специалистов по программированию на языке Java представляет собой обновленное издание фундаментального труда, учитывающее всю специфику новой версии платформы Java SE 6. Подробно рассматриваются такие темы, как организация и настройка среды программирования на Java, фундаментальные структуры данных, объектно-ориентированное программирование и его реализация в Java, интерфейсы, программирование графики, обработка событий, Swing, развертывание приложений и аплетов, отладка, обобщенное программирование, коллекции и построение многопоточных приложений. Книга изобилует множеством примеров, которые не только иллюстрируют концепции, но также демонстрируют способы правильной разработки, применяемые в реальных условиях. Книга рассчитана на программистов разной квалификации, а также будет полезна студентам и преподавателям.

ISBN 978-5-8459-1378-4

в продаже

MICROSOFT WINDOWS SERVER 2008 R2
Полное руководство

Рэнд Моримото,
Майкл Ноэл,
Омар Драуби,
Росс Мистри,
Крис Амарис



www.williamspublishing.com

В книге приводится исчерпывающее описание всех аспектов Windows Server 2008 R2, включая Active Directory, сетевые службы, безопасность, переход на Windows Server 2008 R2 из Windows Server 2003/2008, администрирование, отказоустойчивость, оптимизация, обнаружение неполадок, ключевые прикладные службы и многое другое.

Авторы очень точно отмечают основные усовершенствования Windows Server 2008 R2 и предлагают развернутое описание всех инноваций Windows Server 2008 R2, начиная с виртуализации Hyper-V и заканчивая DirectAccess и улучшениями компонента Failover Clustering. В каждой главе предлагаются многочисленные полезные советы и трюки, основанные на сотнях внедрений, которые позволяют максимально эффективно решать бизнес-задачи с помощью Windows Server 2008 R2.

Книга рассчитана на пользователей и администраторов средней и высокой квалификации.

ISBN 978-5-8459-1653-2 **в продаже**

C# 4.0
ПОЛНОЕ РУКОВОДСТВО

Герберт Шилдт

www.williamspublishing.com

В этом полном руководстве по C# 4.0 — языку программирования, разработанному специально для среды .NET, — детально рассмотрены все основные средства языка: типы данных, операторы, управляющие операторы, классы, интерфейсы, методы, делегаты, индексаторы, события, указатели, обобщения, коллекции, основные библиотеки классов, средства многопоточного программирования и директивы препроцессора. Подробно описаны новые возможности C#, в том числе PLINQ, библиотека TPL, динамический тип данных, а также именованные и необязательные аргументы. Это справочное пособие снабжено массой полезных советов авторитетного автора и сотнями примеров программ с комментариями, благодаря которым они становятся понятными любому читателю независимо от уровня его подготовки. Книга рассчитана на широкий круг читателей, интересующихся программированием на C#.

ISBN 978-5-8459-1684-6

в продаже

BESTSELLING
BOOK
SERIES

Android™ Разработка приложений для чайников®

СЕРИЯ
КОМПЬЮТЕРНЫХ
КНИГ ОТ
ДИАЛЕКТИКИ

Полезные советы по разработке приложений

Некоторые проблемы в кодах Android решаются легко, но часто возникают ситуации, когда непонятно, что делать. Ниже приведены советы, которые помогут вам в работе.

- ✓Если вы не знаете, как решить проблему, посетите форум StackOverflow.com и откройте раздел Android. В этом разделе можете оставить свой вопрос, и вам, скорее всего, ответят люди, которые знают, как решить вашу проблему.
- ✓Готовые решения наиболее распространенных задач можно найти на сайте разработчиков приложений Android:
  `http://d.android.com/guide/appendix/faq/commontasks.html`
- ✓Чтобы узнать, что делает тот или иной класс или пакет, перейдите рабочую среду Eclipse, откройте редактор кода Java и наведите указатель на нужный объект. Появится всплывающее окно с короткой подсказкой. Если этой подсказки недостаточно, просмотрите официальное описание соответствующего класса Android:
  `http://d.android.com/reference/classes.html`
- ✓Чтобы увидеть все ссылки на конкретный объект в файле класса, выделите объект и нажмите комбинацию клавиш <Ctrl+Shift+G>.
- ✓Чтобы сделать поток сообщений в окне DDMS более удобным, создайте фильтр, пропускающий только сообщения, имеющие отношение к вашему приложению.
- ✓Вводя код в окне редактора Eclipse, вы знаете имя целевого свойства, метода или класса, который хотите создать, но в классе его пока что не существует. Введите имя в поле компонентов, и Eclipse сообщит вам, что такой компонент не найден. Выделите имя и нажмите клавишу <F2>. Появится всплывающее окно, в котором можно создать компонент простым щелчком на кнопке.
- ✓Чтобы быстро найти нужный член класса, нажмите комбинацию клавиш <Ctrl+O> и начните вводить имя. Будет отображена всплывающая подсказка со списком имен. Выберите нужное имя и нажмите клавишу <Enter>.
- ✓Для быстрого перебора вкладок в рабочей среде Eclipse нажимайте комбинации клавиш <Shift+PgUp> и <Shift+PgDn>.
- ✓Чтобы запустить приложение Android на выполнение, нажмите комбинацию клавиш <Ctrl+Shift+F11>.

Быстрые клавиши Eclipse

Операция	***Комбинация клавиш***
Создание нового файла в текущем пакете	<Alt+Shift+N>
Упорядочение команд импорта	<Ctrl+Shift+O>
Переход к определениям сущностей	<F3>
Переименование объекта	<Alt+Shift+R>
Поиск в файлах Java	<Ctrl+H>
Открытие сущности определенного типа	<Ctrl+Shift+T>
Поиск объявления объекта	<Ctrl+G>
Переход влево	<Alt+←>
Переход вправо	<Alt+→>

BESTSELLING
BOOK
SERIES

Android™ Разработка приложений для чайников®

СЕРИЯ
КОМПЬЮТЕРНЫХ
КНИГ ОТ
ДИАЛЕКТИКИ

Шпаргалка

Синтаксис намерений Android

Намерение	Пример кода
Запуск деятельности	`startActivity(new Intent(this, Destination.class));`
Создание окна выбора	`Intent.createChooser(yourIntent, "Выберите нужный пункт");`

Открытие веб-браузера	`Intent i=new;`
	`Intent(Intent.ACTION_VIEW, Uri.parse("http://example.org"));`
	`startActivity(i);`

Запуск деятельности с результатом	`startActivityForResult(yourIntent,YOUR_REQUEST_CODE);`

Разработка приложений для разных разрешений экрана

✓Требования к размерам значков при разных разрешениях экрана зависят от типа и местоположения значка. Значки могут быть расположены на панели запуска, в меню, в строке состояния, на корешках вкладок и в других местах. Для каждого разрешения экрана значки прорисовываются по-разному. Ознакомьтесь с учебниками по созданию значков.

✓При определении элементов пользовательского интерфейса рекомендуется всегда использовать единицы измерения **dip** (density-independent pixels — пиксели, не зависящие от разрешающей способности). Тогда приложение будет плавно масштабироваться для разных размеров устройства. Единицы **dip** — это виртуальные пиксели, которые масштабируются пропорционально размеру экрана.

✓Вставляйте в файл манифеста **AndroidManifest.xml** элемент supports-screens, помогающий сайту **Android Market** правильно определить совместимость вашего приложения с экранами разных размеров.

✓Создавайте графику отдельно для экранов с высокой, средней и низкой разрешающими способностями. Время разработки из-за этого увеличится, но повышение эффективности работы и улучшение внешнего вида пользовательского интерфейса с лихвой компенсируют затраты времени.