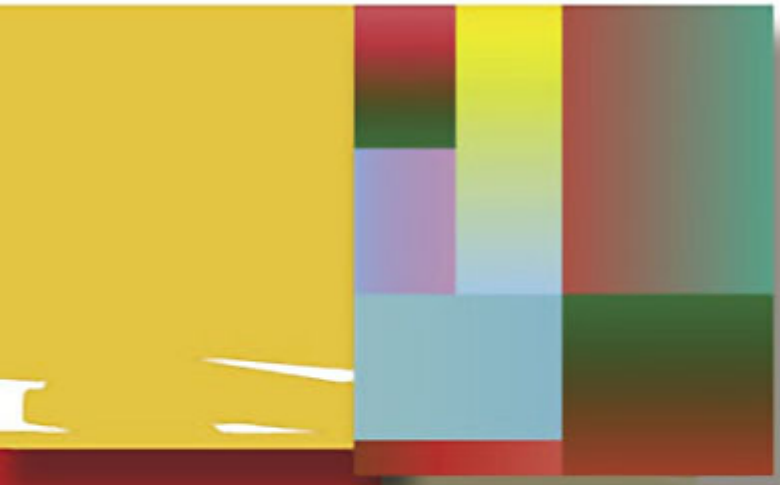
АЛЕКСЕЙ ГОЛОЩАПОВ

bhv®

Google Android

системные компоненты и сетевые коммуникации



ДОСТУП К СИСТЕМНЫМ КОМПОНЕНТАМ

ЛОКАЛЬНЫЕ СЛУЖБЫ

СЕТЕВЫЕ КОММУНИКАЦИИ

СЕРВИСЫ GOOGLE

РАБОТА С ОБОРУДОВАНИЕМ МОБИЛЬНОГО ТЕЛЕФОНА

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Материалы на www.bhv.ru

УДК 681.3.06
ББК 32.973.26-018.2
Г61

Голощапов А. Л.

Г61 Google Android: системные компоненты и сетевые коммуникации. — СПб.: БХВ-Петербург, 2012. — 384 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0666-3

Книга посвящена разработке программ для мобильных устройств под управлением операционной системы Android. Рассматривается создание приложений с использованием системных компонентов и служб Android, управление сетевыми соединениями и коммуникация через сотовую сеть, мобильный Интернет, Wi-Fi. Описана работа с оборудованием мобильного устройства Android: встроенными датчиками, картой памяти, видеокамерой, дисплеем, управление энергопотреблением телефона. Показано использование сетевых сервисов Google в пользовательских приложениях: определение координат, навигация, Geocoding, карты Google Map. Рассматриваемые в книге примеры приложений можно скачать по ссылке: ftp://85.249.45.166/9785977506663.zip и на странице книги на сайте www.bhv.ru.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Нина Седых*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

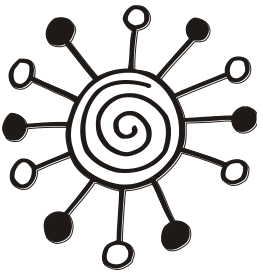
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.11.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 30,96.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-5-9775-0666-3

Оглавление



Глава 1

Подключение мобильного устройства для тестирования и отладки приложений

Прежде чем мы приступим к разработке приложений, представленных в этой книге, я бы хотел обратить ваше внимание на инструменты, требуемые для разработки приложений под Android. В этой главе описываются настройка и подключение мобильного телефона Android для тестирования и отладки приложений, которые мы будем разрабатывать в процессе работы с книгой.

Среда разработки и Android SDK

Поскольку вы уже наверняка знакомы с разработкой приложений под Android, на вашем рабочем компьютере должны быть установлены необходимые инструменты. Для работы нам понадобятся последняя версия среды разработки Eclipse (на момент написания книги — Eclipse Helios 3.6.2) и плагин ADT (Android Developer Tools) для нее. Версия плагина ADT — 10.0.1 или выше, если к моменту, когда вы будете читать эту книгу, выйдет следующая.

Также для работы с книгой у вас должны быть установлены библиотеки Android SDK версии 2.3.3 (API Level 10). Версия Android SDK 3.0 (API Level 11) не обязательно должна присутствовать в вашей инсталляции, поскольку мы будем рассматривать создание приложений только для смартфонов Android, а работа с планшетными PC — это тема для другой книги.

Если вы давно не обновляли свою инсталляцию Android SDK, проверьте установленные у вас библиотеки. Откройте SDK Manager в каталоге, где у вас установлен Android SDK, и в окне **Android SDK and AVD Manager** откройте узел **Installed Packages**. На правой панели отобразятся установленные пакеты, как показано на рис. 1.1.

Кроме Android SDK версии 2.3.3, нам понадобятся библиотеки Google API для работы с сетевыми сервисами Google. Эти библиотеки не входят в состав Android SDK и устанавливаются отдельно. У вас должны быть инсталлированы библиотеки до Google API Level 10 включительно (рис. 1.1).

Если у вас не установлены требуемые библиотеки, обязательно установите их, они понадобятся нам для работы с примерами приложений из этой книги. Следующий шаг, который необходимо сделать, — заняться настройкой мобильного телефона.

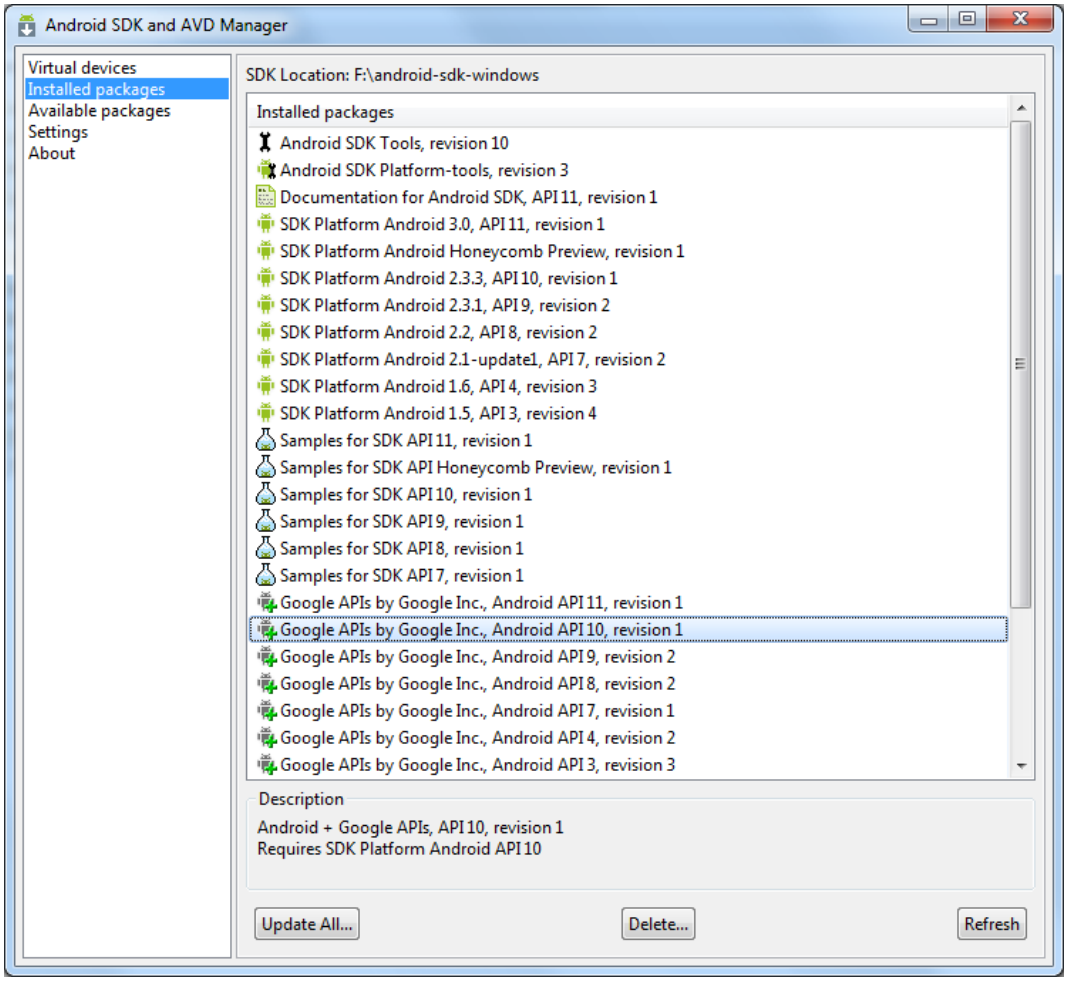


Рис. 1.1. Установленные пакеты Android SDK и Google API

Настройка мобильного телефона для отладки приложений

Эмулятор Android предоставляет широкие возможности для отладки и тестирования приложений. Те читатели, кто уже изучил мою предыдущую книгу [1], посвященную базовым принципам программирования на платформе Android, уже познакомились с функциональностью, предоставляемой эмулятором мобильного устройства Adnroid. Практически все приложения для платформы Android, создаваемые в книге [1], — службы, приложения с графическим интерфейсом, приложения для работы с базами данных, графикой и мультимедиа — позволяют использовать эмулятор Android для отладки и тестирования.

Однако при разработке коммерческого приложения для мобильного устройства Android необходимо проверить свое приложение на реальном телефоне, прежде чем передавать его пользователям. Кроме того, необходимо учитывать, что не все приложения можно отладить на эмуляторе Android. Безусловно, в эмуляторе Android имеется множество возможностей, однако некоторую функциональность требуется отлаживать и тестировать только на реальном устройстве. Например, запускать приложения, работающие по Wi-Fi, Bluetooth, использующие встроенные в мобильный телефон сенсоры для измерения температуры, ориентации телефона, уровня зарядки батареи, а также встроенную видеокамеру, необходимо только на реальном мобильном устройстве.

На реальном мобильном устройстве Android можно тестировать и отлаживать создаваемые приложения так же, как и на эмуляторе Android. Для того чтобы сконфигурировать мобильное устройство для инсталляции, запуска и отладки приложений, необходимо выполнить несколько шагов:

- ❒ Установить на мобильном телефоне режим отладки через USB.
- ❒ Установить на компьютере, где находится среда разработки, драйвер USB для вашего мобильного устройства, если вы до сих пор его не поставили.
- ❒ Проверить работоспособность соединения с помощью инструмента DDMS (Dalvik Debug Monitor Server).

Установка режима отладки через USB

На мобильном телефоне выберите **Settings** (обычно эта опция выводится на домашнем экране Home Screen, если нет — ее можно найти в меню). Затем последовательно выберите **Applications | Development | USB debugging** и поставьте флажок **Debug mode when USB is connected**, как показано на рис. 1.2.

После того как вы выполнили эти действия, подключите ваш мобильный телефон к компьютеру, используя кабель USB, который, как правило, всегда идет в комплекте с мобильным телефоном.

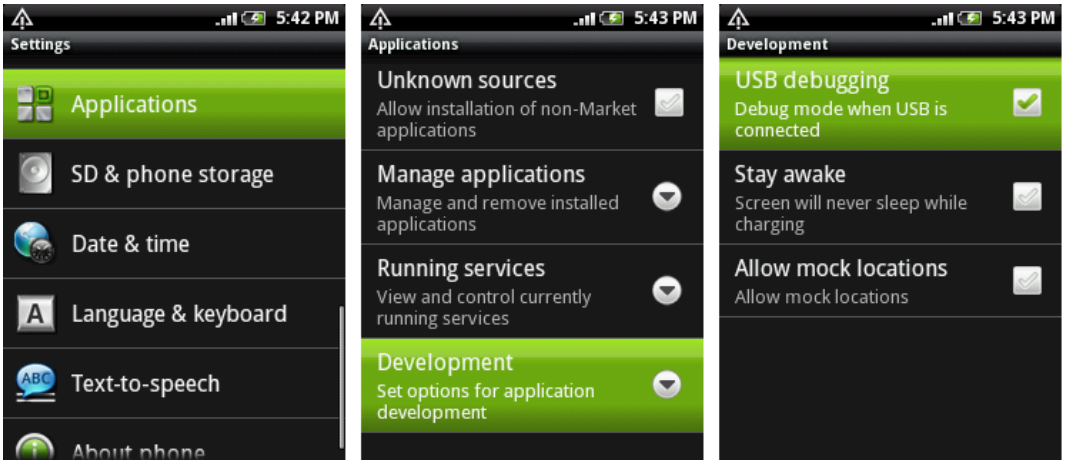


Рис. 1.2. Включение режима USB debugging на мобильном телефоне

Установка драйвера USB

Чтобы соединить мобильный телефон с вашим компьютером, на котором установлена среда разработки, потребуется инсталляция USB-драйвера, совместимого с моделью вашего телефона. Здесь возможны два варианта:

- использовать драйверы USB, поставляемые в составе Android SDK;
- использовать драйвер USB, предоставляемый производителем данной модели мобильного телефона.

Драйвер USB для Windows доступен для загрузки как дополнительный компонент Android SDK. Драйверы USB для мобильного телефона, поставляемые вместе с Android SDK, расположены в каталоге android-sdk-windows\google-usb_driver.

Однако эти драйверы рассчитаны только на несколько типов моделей телефонов, поэтому лучше выбрать второй вариант и инсталлировать драйвер USB, предоставляемый производителем вашего мобильного телефона. Обычно все производители предлагают отдельные драйверы или пакеты для синхронизации мобильного устройства и компьютера, в комплект которых уже входит драйвер USB.

Процесс установки драйвера USB очень простой, и я не буду его здесь рассматривать, наверняка вы все умеете это делать.

Взаимодействие мобильного телефона с DDMS

После подключения телефона и установки драйвера USB необходимо проверить взаимодействие мобильного устройства и инструментов для отладки Android-приложений. Для этого запустите инструмент Dalvik Debug Monitor Server из подкаталога tools в каталоге установки вашего Android SDK или напрямую из IDE Eclipse, выбрав представление **Devices** (пункт меню **Window | Show View | Other | Android | Devices**).

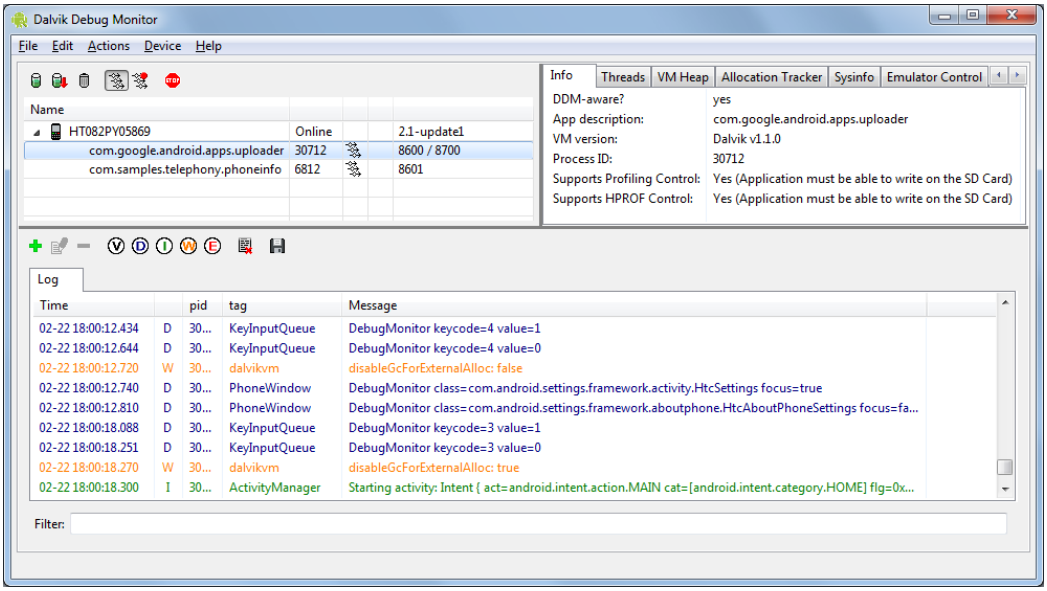


Рис. 1.3. Подключение мобильного устройства к DDMS

Если все было правильно установлено и настроено, в окне **Dalvik Debug Monitor** на панели **Name** будет отображаться подключенное внешнее мобильное устройство, как показано на рис. 1.3.

Инструмент DDMS работает с реальным мобильным устройством так же, как и с эмулятором Android. Вы можете получать информацию о запущенных процессах, системных событиях, имеете доступ к файловой системе и многим другим функциям, предоставляемым этим инструментом.

Запуск и инсталляция проекта на мобильный телефон из IDE Eclipse

Теперь мобильное устройство успешно подключено к компьютеру, в среде Eclipse можно выбирать место развертывания вашего приложения: эмулятор Android или мобильное устройство. При запуске проекта на выполнение появится диалоговое окно **Android Device Chooser**, которое позволяет выбрать место для развертывания приложения (рис. 1.4), при условии, что у вас запущен эмулятор Android. Если эмулятор не запущен, окно выбора устройства не появится и приложение развернется сразу на мобильном устройстве.

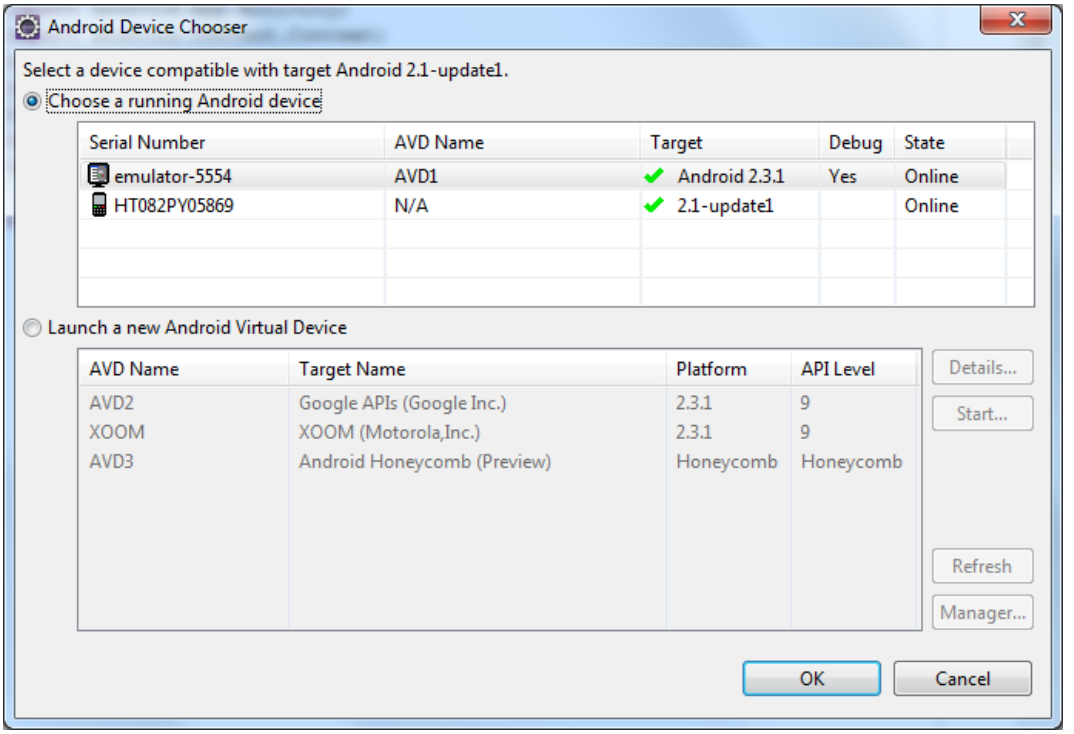


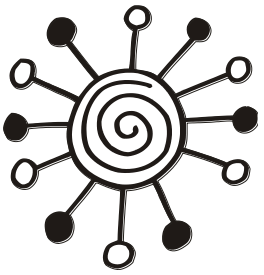
Рис. 1.4. Выбор целевого устройства для инсталляции и запуска проекта

Резюме

В этой главе мы рассмотрели подключение и настройку мобильного устройства на платформе Android для тестирования и отладки приложений, необходимые для работы с примерами, предоставленными в книге. Однако если у вас пока нет в наличии телефона Android, не расстраивайтесь. Многие примеры приложений в этой книге позволяют обойтись эмулятором Android.

Но необходимо учитывать, что для разработки приложений, использующих соединения Wi-Fi (*глава 7*), встроенную видеокамеру (*глава 12*), аппаратные датчики (*глава 13*), доступ к аккумуляторной батарее (*глава 15*) и управление энергопотреблением (*глава 16*), без мобильного телефона Android не обойтись.

В следующей главе будут рассмотрены базовые принципы вызова локальных и сетевых служб, системных компонентов и стандартных окон для настройки внутренних баз данных, а также установка требуемых разрешений для обеспечения приложениям доступа к системным компонентам Android и сетевым коммуникациям.



Глава 2

Доступ к системным компонентам и сетевым сервисам

Прежде чем мы перейдем непосредственно к созданию приложений с использованием функциональностей, предоставляемых системными компонентами Android, имеет смысл сделать небольшой обзор этих компонентов и обсудить некоторые базовые принципы работы с ними.

В этой главе описываются общие принципы организации системных компонентов Android и доступа к ним приложений.

Компоненты системы Android

Те из читателей, кто уже изучал самостоятельно архитектуру системы Android, должны быть знакомы с базовыми компонентами Android. Их всего четыре:

- Activity;
- Service;
- Broadcast Receiver;
- Content Provider.

Из моей книги [1] вы уже узнали принципы работы с компонентами, научились строить приложения с графическим интерфейсом, создавать и вызывать собственные службы, работать с базами данных. Система Android — это многофункциональная платформа, внутри нее уже реализованы многочисленные компоненты, которые можно использовать в ваших приложениях. Поэтому при разработке собственных приложений нет необходимости заново изобретать велосипед, а лучше по возможности использовать функциональность существующих компонентов системы Android.

Не стоит также забывать и тот факт, что мобильный телефон является в первую очередь средством коммуникации и обладает, помимо обеспечения связи между абонентами сотовой сети, еще и доступом к Интернету. Это означает, что в ваших приложениях можно использовать различные удаленные сетевые службы, предоставляемые Google, например, Google Maps, Street View и др.

Системные службы

Платформа Android предоставляет в ваше распоряжение множество служб, функциональность которых вы можете использовать в разрабатываемых приложениях. Количество служб с выходом каждой новой версии постоянно увеличивается, и на момент выхода версии 3.0 их уже более 35.

Мы кратко рассмотрим доступные в системе Android службы и их предназначение, чтобы вы могли получить полное представление о возможностях служб, входящих в состав платформы Android.

Мобильный телефон Android предоставляет пользователю удобный и красивый графический интерфейс. Для его поддержки в системе существует группа служб для управления графическим пользовательским интерфейсом:

- *Wallpaper Service* — служба для управления "обоями", загружаемыми на Home Screen (домашний экран) мобильного устройства;
- *Layout Inflater Service* — служба для управления компоновкой окон при их динамической загрузке;
- *UI Mode Service* — служба, предназначенная для управления режимами работы пользовательского интерфейса.

Для организации взаимодействия пользователя с мобильным устройством в состав платформы Android включены службы, которые обеспечивают мониторинг и управление пользовательским вводом:

- *Input Method Service* — служба, используемая для текстового ввода. Эта служба обеспечивает отображение автодополнения при вводе текста пользователем в текстовое поле. Например, эта служба работает, когда вы вводите текст SMS-сообщения;
- *Accessibility Service* — обеспечивает доступ к событиям, возникающим в пользовательском интерфейсе, например, при нажатии пользователем на кнопку, получению или потери фокуса виджетом;
- *Clipboard Service* — управляет буфером обмена. Буфер обмена в Android, в отличие от других систем, может использоваться только для текста и является глобальным, т. е. виден для всей системы, а не для отдельного приложения;
- *Search Service* — служба для управления функциями глобального поиска на устройстве;
- *Keyguard Service* — служба для управления блокировкой клавиатуры мобильного устройства.

Важную роль в платформе Android играет группа системных служб, предназначенных для организации взаимодействия компонентов операционной системы и приложений, ведения журнала системных событий и оповещения пользователя о событиях, происходящих в системе:

- *Notification Service* — управляет пользовательскими уведомлениями (например, при получении SMS-сообщения), которые отображаются в строке состояния мобильного устройства;
- *Alarm Service* — предназначена для отправления пользователю разовых или периодических оповещений в заданное время в виде звукового сигнала (например, будильник) или световых сигналов, виброзвонка, тестовых сообщений;
- *Window Service* — служба для доступа к системному менеджеру окон;
- *Activity Service* — служба для взаимодействия с объектами Activity, открытыми в данный момент времени в системе. Кстати, служба Activity Service может взаимодействовать не только с Activity, но и с другими службами, находящимися в системе;

- *Dropbox Service* — предназначена для обеспечения записи системных событий в диагностический файл.

Для управления медиа — записью и воспроизведением музыки и видео, громкостью, режимами телефонных звонков — на платформе Android предусмотрена служба *Audio Service*.

Для работы с оборудованием на платформе Android также существует набор служб, которые обеспечивают доступ к "железу" мобильного телефона и использование его в приложениях:

- *Power Service* — служба для управления энергопотреблением. Эта служба предназначена в первую очередь для эффективного использования батареи на мобильном устройстве. Она позволяет управлять энергопотреблением процессора, подсветкой экрана и клавиатуры (если таковая имеется на телефоне);
- *Battery Manager* — служба для контроля состояния аккумуляторной батареи. Эта служба получает данные о степени заряда батареи, ее температуре, подключении зарядного устройства и другую важную информацию об источнике питания мобильного устройства;
- *Sensor Service* — служба для доступа к встроенным датчикам, например, акселерометру, датчику температуры, освещенности мобильного устройства;
- *Storage Service* — служба для управления сохранением данных в файловой системе: во внутренней памяти устройства и на съемной карте памяти;
- *Vibrator Service* — служба, обеспечивающая доступ и управление виброзвонком мобильного устройства.

Поскольку основным назначением мобильного устройства является обеспечение коммуникации, система Android располагает большим количеством служб для управления сетевыми соединениями и передачей данных:

- *Telephony Service* — служба для управления телефонными функциями мобильного устройства. Эта служба управляем телефонными вызовами, отправкой и приемом SMS-сообщений;
- *Connectivity Service* — служба для управления сетевыми соединениями мобильного телефона;
- *Download Service* — служба для управления загрузками данных через HTTP;
- *Wifi Service* — служба для управления сетевым соединением Wi-Fi;
- *Location Service* — служба для отслеживания физических координат местоположения мобильного устройства. Эта служба использует провайдеров местоположения и сетевые сервисы Google;
- *NFC Service* — служба для управления NFC (Near Field Communication). Это технология беспроводной высокочастотной связи малого радиуса действия, позволяющая производить соединение и обмен данными между устройствами, находящимися на расстоянии нескольких сантиметров. Эта новая технология, предназначенная для мобильных телефонов и платежных систем.

В мобильном устройстве Android также предусмотрен набор служб для управления безопасностью, учетными записями и политиками безопасности:

- *Account Service* — используется для управления пользовательскими учетными записями, предназначенными для доступа к различным онлайн-сервисам, например Gmail, Facebook и др.;

- *Device_policy Service* — предназначена для администрирования устройства и управления пользовательскими политиками безопасности, например, устанавливает минимальную длину пароля, время действия пароля и др.

Для доступа к службам в приложениях в Android SDK используются менеджеры служб. У каждой службы, за редким исключением, есть собственный менеджер. Так, например, для управления службой *Power Service* используется Power Manager, для *Telephony Service* — Telephony Manager и т. д.

Подробно все службы платформы Android мы рассмотреть не можем ввиду ограниченного объема книги, но применению большинства из перечисленных служб, которые могут наиболее часто использоваться в приложениях, мы уделим достаточно внимания в следующих главах.

Объекты *Intent*

Помимо служб, перечисленных в предыдущем разделе, в системе Android существует множество стандартных Activity, которые вы можете запускать из своего приложения, используя объекты `Intent`. Эти объекты в системе Android служат средством для позднего связывания во время выполнения между компонентами одного или нескольких приложений.

Все объекты `Intent` можно разделить на две категории:

- *Activity Action Intent* — используются для вызова Activity, находящегося в другом приложении. Для вызова абонента и ответа на входящий звонок можно вызвать соответствующий Activity. Если нам требуется включить сетевое соединение, можно сделать элемент управления (например, кнопку) для включения соединения в приложении или вызвать стандартный системный Activity, который предоставляет разнообразные опции для управления и конфигурирования сетевого соединения;
- *Broadcast Intent* — рассылаются всем компонентам, находящимся на мобильном устройстве для оповещения о каком-либо событии, произошедшем в системе. Например, при получении мобильным устройством входящего SMS-сообщения система Android посылает оповещение Broadcast Intent. Любое приложение может перехватить и обработать этот объект `Intent`, отреагировав на это событие соответствующим образом.

Intent-фильтры

Некоторые оповещения Broadcast Intent посылаются системой с большой частотой, например, событие таймера. Другие объекты Broadcast Intent, которые, например, генерируются при изменении уровня сигнала сетевого соединения, могут изменять частоту своих рассылок в зависимости от внешних условий. При неустойчивой связи или быстрых перемещениях мобильного телефона вместе с пользователем частота генерации таких оповещений будет намного больше, чем при неподвижном положении мобильного телефона.

Поэтому при использовании объектов Broadcast Intent в приложениях желательно устанавливать фильтры, чтобы не получать оповещений, которые не требуются

для работы вашего приложения. Для этого существуют специальные фильтры объектов `Intent` (Intent Filter). Мы их также будем применять, в случае необходимости, при создании приложений в этой книге.

Системные контент-провайдеры

Доступ к данным и работа с данными в приложениях в системе Android осуществляется через следующие механизмы:

- база данных SQLite — реляционная база данных, которая используется на платформе Android;
- контент-провайдеры — компоненты системы, которые предоставляют приложениям интерфейс для работы с данными.

Контент-провайдеры сохраняют и возвращают данные для приложения, в котором они используются. По сути, они являются посредниками между приложением и базой данных.

Встроенные базы данных

Платформа Android предоставляет несколько встроенных баз данных, доступных для приложений через соответствующие контент-провайдеры.

Вы можете получить доступ к этим контент-провайдерам, используя функциональность, предоставляемую пакетом `android.provider`. Это стандартные контент-провайдеры, которые всегда присутствуют на мобильном телефоне Android, независимо от производителя и модели:

- *ContactsContract* — используется для сохранения данных пользовательских контактов в телефоне;
- *CallLog* — используется для сохранения всех вызовов: входящих и исходящих звонков, пропущенных вызовов и продолжительности звонка;
- *Browser* — используется для сохранения истории просмотренных страниц, созданных закладок и поиска;
- *Mediastore* — используется для сохранения и организации доступа к медиафайлам и коллекциям: музыки, видео и фотографиям;
- *Settings* — используется для сохранения пользовательских настроек мобильного телефона, например, языка интерфейса, яркости экрана, уровня громкости звонка, конфигурации сети и т. д.;
- *UserDictionary* — используется для сохранения пользовательских словарей, которые применяются для автодополнения при вводе текста.

Каждому из перечисленных контент-провайдеров в системе Android соответствует одноименная база данных.

Существует также база данных для хранения SMS-сообщений. По какой-то причине документация по базе данных SMS отсутствует в официальной документации Android SDK, однако мы ее рассмотрим в *главе 5*, при изучении способов программного перехвата входящих и исходящих SMS-сообщений.

Кроме перечисленных, на мобильном телефоне также имеются различные специализированные базы данных, предназначенные для обеспечения работы различных приложений. Например, веб-браузер, помимо базы данных Browser, использует дополнительную базу данных WebviewCache для кэширования, будильник использует собственную базу данных Alarms.

Установка требуемых разрешений в приложении

По умолчанию, доступ почти ко всем системным компонентам ограничен для внешних приложений. Поэтому если приложению требуется использовать какую-либо системную службу, например, необходимо послать сообщение SMS, подключиться к мобильному Интернету или к удаленному сетевому сервису, нужно задать для приложения соответствующие разрешения.

При создании приложений в этой книге мы будем постоянно добавлять требуемые разрешения в файл манифеста приложения. Для тех читателей, кому еще не доводилось применять разрешения в своих приложениях, я кратко расскажу об их использовании.

Разрешение можно задать напрямую в коде файла манифеста приложения, добавив элемент `<uses-permission>` с атрибутом, содержащим имя нужного разрешения, или, что более удобно (поскольку при редактировании файла манифеста не работают всплывающие подсказки, а разрешений очень много), использовать для этого окно редактора файла манифеста, а именно его вкладку **Permissions**. На этой вкладке нажмите кнопку **Add** и в появившемся диалоговом окне выберите элемент **User Permissions**, как показано на рис. 2.1.

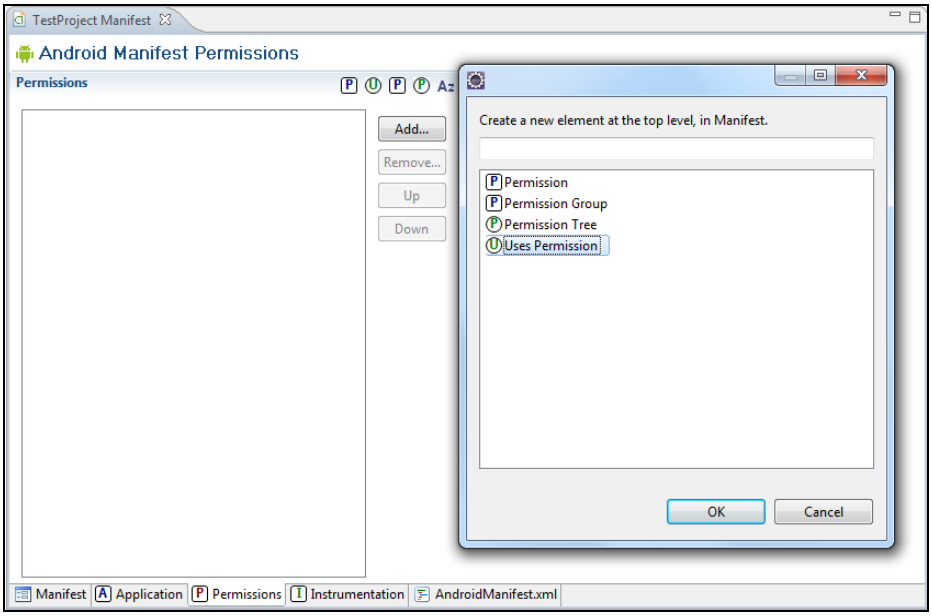


Рис. 2.1. Установка разрешений в файле AndroidManifest.xml

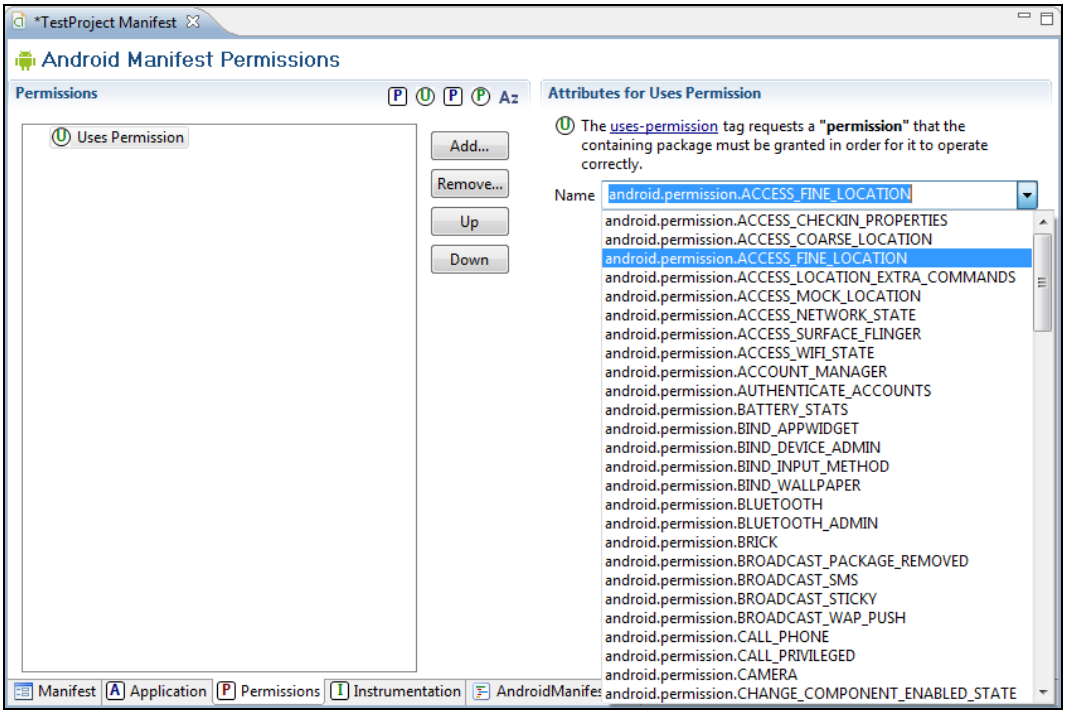


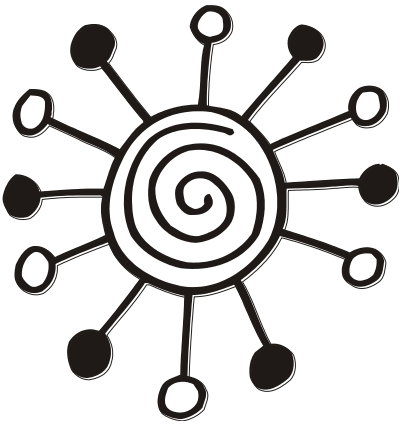
Рис. 2.2. Выбор необходимого разрешения для доступа к системным компонентам

После этого на вкладке **Permissions** в правой части появится панель **Attributes for Uses Permission**. На этой панели в выпадающем списке **Name** находится полный список всех доступных разрешений (рис. 2.2), которые вы можете использовать у себя в приложении. После выбора нужного разрешения редактор манифеста добавит его в файл AndroidManifest.xml.

При проектировании приложения программисты довольно часто забывают подключить разрешения. Впрочем, это сразу обнаруживается при запуске приложения — при отсутствии нужного разрешения генерируется исключение.

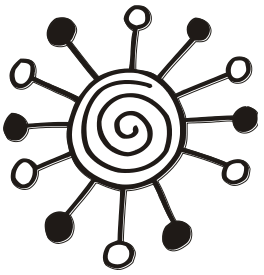
Резюме

В этой главе были рассмотрены компоненты системы Android и общая организация доступа к системным компонентам Android, внутренним базам данных и сетевым службам. Более детализированный доступ к системной функциональности из кода приложений и работа с компонентами, входящими в состав платформы Android, будут рассматриваться в следующих главах книги.



ЧАСТЬ II

Базовые функции телефона и сетей сотовой связи



Глава 3

Получение информации о телефоне и сети сотовой связи

В этой главе будет рассказано о том, как получить информацию о телефоне и сети сотовой связи и использовать ее в своих приложениях. Система Android позволяет установленным на мобильном устройстве приложениям производить телефонные вызовы, отправлять и принимать SMS-сообщения, отслеживать состояние телефона и сети сотовой связи. Весь этот богатый набор средств управления телефоном вы можете использовать при разработке собственных приложений.

Информация о телефоне

Для доступа к информации о телефоне и сети сотовой связи используется класс `TelephonyManager`, который находится в пакете `android.telephony`. Это класс позволяет приложению получить доступ к системной службе мобильной телефонии на устройстве.

Экземпляр этого класса не создается напрямую в программе. Для получения экземпляра `TelephonyManager` необходимо вызвать метод `getSystemService()`, передав ему параметр `Context.TELEPHONY_SERVICE`:

```
TelephonyManager manage = (TelephonyManager)getSystemService(
        Context.TELEPHONY_SERVICE);
```

Класс `TelephonyManager` имеет большой набор методов для определения типа и состояния телефона. Кроме того, с помощью этих методов вы можете в своем приложении определять тип и доступность сети сотовой связи, состояние SIM-карты и многое другое. Далее в этой главе мы подробно разберем использование фунциональности, предоставляемой классом `TelephonyManager` для управления службой Telephony Service и получения от этой службы нужной нам информации.

Определение типа телефона и сети сотовой связи

Для определения типа телефона и доступности сети сотовой связи в классе `TelephonyManager` есть методы `getPhoneType()` и `getNetworkType()`. Эти методы возвращают целочисленные константы, определяющие типы телефона и сети сотовой связи.

Метод `getPhoneType()` определяет тип мобильного телефона и возвращает одно из четырех значений:

- `PHONE_TYPE_GSM` — телефон стандарта GSM (Global System for Mobile communications, глобальный цифровой стандарт для сотовой связи). Этот стандарт, я думаю, известен всем. Он был разработан Европейским институтом стандартов телекоммуникаций ETSI еще в 80-х годах. Стандарт GSM на сегодняшний день является наиболее распространенным стандартом связи;
- `PHONE_TYPE_CDMA` — CDMA (Code Division Multiple Access, множественный доступ с кодовым разделением). Это технология мобильной связи, при которой каналы передачи имеют общую полосу частот, но разную кодовую модуляцию;
- `PHONE_TYPE_SIP` — SIP (Session Initiation Protocol, протокол инициализации соединения). Несмотря на то, что эта технология известна уже давно и широко используется в IP-телефонии, поддержка SIP в Android появилась совсем недавно, в версии Android SDK 2.3 (API Level 9);
- `PHONE_TYPE_NONE` — это значение используется, если по каким-то причинам тип телефона не подходит под предыдущие определения.

Метод `getNetworkType()`, определяющий тип сотовой сети, в зоне действия которой в данный момент находится мобильный телефон, возвращает гораздо больше вариантов значений, чем метод `getPhoneType()`:

- `NETWORK_TYPE_GPRS` — сеть GPRS (General Packet Radio Service, сеть с пакетной передачей данных). Это расширение технологии GSM для пакетной передачи данных. Технология GPRS позволяет пользователю мобильного телефона производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе с Интернетом. Весь поток данных отправителя разбивается на отдельные пакеты и затем доставляется получателю. Технология GPRS позволяет передавать данные на существенно более высоких скоростях, чем в обычной GSM-сети;
- `NETWORK_TYPE_CDMA` — сеть CDMA (эта технология уже была описана ранее в этом разделе);
- `NETWORK_TYPE_1xRTT` — сеть 1xRTT (One Times Radio Transmission Technology). Это мобильная технология передачи цифровых данных, основанная на CDMA-технологии, но использующая принцип передачи с коммутацией пакетов;
- `NETWORK_TYPE_EDGE` — сеть EDGE (Enhanced Data Rates for Global Evolution, технология высокоскоростной передачи данных в сетях GSM). Технология EDGE является расширением технологии передачи данных с коммутацией каналов для увеличения пропускной способности этого сервиса;
- `NETWORK_TYPE_EHRPD` — сеть EHRPD (Evolved High-Rate Packet Data, улучшенная технология высокоскоростной пакетной передачи данных);
- `NETWORK_TYPE_LTE` — сеть LTE (Long Term Evolution). Технология LTE является усовершенствованием технологий CDMA и UMTS с повышенной скоростью передачи данных и уже относится к сетям четвертого поколения (4G);
- `NETWORK_TYPE_EVDO_0`, `NETWORK_TYPE_EVDO_A`, `NETWORK_TYPE_EVDO_B` — сети типа EVDO (Evolution-Data Optimized). Это технология передачи данных, используемая в сетях сотовой связи стандарта CDMA.1X;