
Программирование приложений
для мобильных устройств под
управлением Android
ANDROID

Евгений Владимирович Сенько

Программирование приложений для мобильных устройств под управлением Android. Часть 1

http://www.litres.ru/pages/biblio_book/?art=11643376

Аннотация

Книга посвящена разработке программ для мобильных устройств под управлением операционной системы Android. Рассматривается создание приложений с использованием системных компонентов и служб Android. Приведены базовые данные о структуре приложений, об основных классах и их методах, сопровождаемые примерами кода. Часть 1 содержит шесть глав, описывающих основные принципы создания приложений, пользовательский интерфейс, полномочия приложений, а так же базовые классы: Activity, Intent, Fragment. Книга предназначена для программистов, владеющих языком программирования Java и желающих освоить написание приложений, работающих под ОС Android. Книга является переводом общедоступных бесплатных англоязычных интернет ресурсов. Во второй части книги будут рассмотрены

Нотификации (Notifications), Broadcast Receivers, Потоки и асинхронное выполнение задач (Threads & AsyncTasks), Оповещения (Alarms), работа с сетью, графика и анимация, управление тачем и жестами, управление мультимедией, работа с датчиками, определение местоположения и привязка к картам, управление данными, а также классы ContentProvider и Service.

Содержание

Основные принципы создания приложения 5
Конец ознакомительного фрагмента. 16

Евгений Сенько Программирование приложений для мобильных устройств под управлением Android. Часть 1

Основные принципы создания приложения

Сначала мы поговорим о четырех фундаментальных строительных блоках, из которых строятся все Android-приложения. Эти блоки реализованы в виде Java-классов. И первый из этих строительных блоков – класс Activity. Это основной класс, который видят пользователи при запуске приложения. Activity разработаны с целью обеспечения графического пользовательского интерфейса или GUI, и они позволяют пользователям передавать и получать информацию из приложения.

Остальные три компонента работают «за кулисами», по-

этому они не имеют пользовательских интерфейсов. Эти компоненты включают в себя:

Service для поддержки длительных или работающих в фоновом режиме операций;

Broadcast receiver – для прослушивания и реагирования на события, которые происходят на устройстве;

Content provider, который позволяет нескольким приложениям хранить и обмениваться данными.

Приложения обычно создаются из нескольких взаимодействующих компонентов, которые запускает Android, чтобы все работало как надо. И каждый из этих компонентов выполняет свои функции в экосистеме Android. И, следовательно, имеет свою точку входа и собственный API. Давайте взглянем на каждый из этих компонентов по одному.

Во-первых, давайте поговорим о классе Activity. Этот класс предоставляет графический интерфейс пользователя и обеспечивает взаимодействие с пользователем через этот интерфейс. Как правило, Activity должна поддерживать одно конкретное действие, которое пользователь может сделать. Такое действие, как набор телефонного номера или ввод контактной информации для одного человека, и так далее. Хотя сейчас уже появились устройства с большими экранами, особенно планшеты, и то, что мы называем одним конкретным действием, которое пользователь может сделать, конечно, может измениться. Теперь, в качестве примера Activity, давайте взглянем на приложение «Номерона-

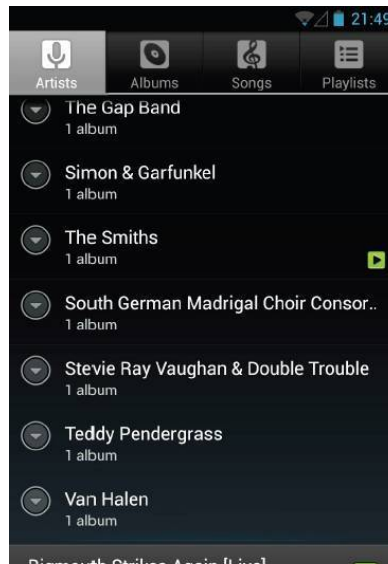
бираТель».

Вы наверняка знаете, что приложение «Телефон» может открыть пользовательский интерфейс с несколькими вкладками. Одна – для набора номера телефона, одна – для списка вызовов, одна – для контактов. В Android 4.2 исходный код этого приложения фактически является частью приложения «Контакты».

Следующий компонент – это класс Service (сервис). В от-

личие от Activity сервисы выполняются в фоновом режиме. Поэтому нет необходимости в пользовательских интерфейсах. Вместо этого сервисы имеют две главные цели. Первое – они могут выполнять длительные операции, как правило, отдельно от основного потока пользовательского интерфейса. И второе – они обеспечивают взаимодействие различных процессов для обмена данными.

Для примера давайте взглянем на приложение «Музыка». Приложение «Музыка» имеет несколько различных экранов пользовательского интерфейса, которые показывают, например, ваши музыкальные записи. Показаны песни в исполнении одного артиста, позволяя вам выбрать одну песню данного исполнителя и проиграть эту песню.



Теперь, если вы запустите воспроизведение песни, но затем решите вернуться и посмотреть, какие еще есть песни данного исполнителя или, если вы хотите сделать что-то совершенно другое, например проверить электронную почту, то вы, вероятно, не захотите чтобы при этом воспроизведение прекратилось. Android обеспечит это с помощью сервиса, чтобы музыка продолжала играть, пока вы проверяете почту.

Следующий компонент – это Broadcast receiver (приемник

сообщений и событий). Broadcast receiver «слушает» и реагирует на события. И, по сути, он играет роль подписчика в модели «публикация-подписка».

В Android события представлены классом Intent. О нем мы поговорим подробнее немного позже. «Издатели» создают этот Intent и затем рассылают его, используя специальные методы, как бы отправляя в эфир. Далее его принимает Broadcast receiver, который и подписан на этот конкретный Intent. И, приняв Intent, он может отреагировать на произошедшее событие.

Пример приложения, которое использует Broadcast receiver – приложение для обмена сообщениями. Давайте представим, что кто-то хочет отправить мне SMS-сообщение. SMS-сообщение будет создано и отправлено через телефонную сеть и в конечном счете достигнет моего телефона. И когда это произойдет, Android выставит значок уведомления в панели уведомлений, который даст мне знать, что для меня пришло сообщение. Но вы, конечно, никогда не можете точно знать, в какой момент вы получите такое сообщение. Поэтому Android имеет некоторое программное обеспечение, которое просто «сидит» и «ждет» когда придет SMS-сообщение. И когда оно приходит, это программное обеспечение передает в эфир Intent SMS_received. И есть еще один Broadcast receiver, который слушает этот Intent SMS_received, и он, получив этот Intent, запустит сервис, который загрузит и сохранит входящее SMS-сообщение.

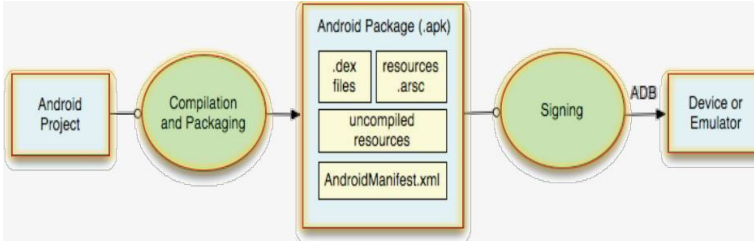
Последний компонент – это Content provider. Content provider позволяет приложениям хранить и обмениваться данными. Content provider использует интерфейс, подобный базам данных, но он больше, чем просто база данных. Например, Content provider сможет обрабатывать связи между процессами. Так что приложения, работающие в отдельных процессах, могут взаимодействовать и обмениваться данными безопасно и легко. Приложение «Браузер» является одним из примеров приложений, использующих Content provider. Если запустить «Браузер» и нажать на значок рядом с адресной строкой в браузере, откроется список закладок и сохраненных адресов веб-сайтов для быстрого доступа в будущем. Когда пользователь добавляет одну из этих закладок, браузер сохраняет ее в Content provider.

Если посмотреть на простейшее приложение «Hello Android», мы увидим всего одну Activity. Но мы будем рассматривать более сложные приложения, в нашем случае, включающее в себя две Activity. И, конечно, как вы понимаете, если можно использовать две Activity, то можно добавить и третью, и четвертую и так далее – просто будет больше одной.

Например, приложение «MapLocation» одним из видов Activity позволяет пользователю ввести почтовый адрес. Эта Activity имеет кнопку, которую пользователь может нажать, как только он ввёл адрес. И, нажав на эту кнопку, запустить вторую Activity – «Google Maps», которая представляет со-

бой карту, отцентрированную на адрес, который пользователь только что ввёл.

Следующая схема показывает процесс написания и построения приложений под Android.



Во-первых, вы создаете исходный код и код ресурсов, которые составляют приложение. Далее, вы компилируете исходный код и подготавливаете свои ресурсы. Результатом этого шага является пакет Android или APK, который является исполняемым файлом приложения. Далее APK подписывается цифровой подписью, чтобы идентифицировать вас как разработчика и, наконец, APK устанавливается на устройство или эмулятор и запускается.

Ваше участие в процессе сборки как разработчика, как правило, включает следующие четыре шага:

- определение ресурсов;
- реализация классов приложения;
- упаковка приложения;
- установка и запуск приложения, в частности для тести-

рования и отладки.

Шаг 1, определение ресурсов приложения. Приложения для Android больше, чем просто исходный код. Они включают в себя исходники, такие как файлы лейаута (layout – размещение наэкранных элементов: кнопок, полей ввода и т. д.), строковых констант, изображений, меню, анимации и многое другое. Управление ресурсами отдельно от приложения имеет несколько преимуществ, одно из которых заключается в том, что вы можете легко изменить эти ресурсы без изменения и перекомпиляции исходного кода приложения.

Один распространенный тип ресурсов – strings (строки). В Android существует три типа строковых ресурсов. Отдельные строки – strings, массивы строк – arrays и формы множественного числа – plurals. Строки и массивы строк довольно понятны, давайте поговорим о plurals. Plurals в основном – это массивы строк, которые могут использоваться, чтобы выбрать определенные строки, связанные с определенной величиной (количеством), например, одна книга или две книги.

Строки обычно хранятся в xml-файлах в каталоге приложения res/values. Строка может включать форматирование и информацию о стилях. Такие вещи, как HTML-теги, например. Основным преимуществом хранения строковых констант в файле ресурсов является простота интернационализации Android-приложений, т. е. их перевод на иностранные языки. Если вы хотите создать приложение, переведенное на несколько разных языков, то рекомендуется выбрать

язык по умолчанию – английский и поместить строки на английском языке в xml-файл strings в папку ресурсов, используемую по умолчанию – res/values. А файлы со строками на других языках – в соответствующие папки, например, для русского языка – в res/values-ru, для французского – в res/values-fr, немецкого – в res/values-de и так далее.

strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="show_map_string">Show Map</string>
    <string name="location_string">Enter Location</string>
</resources>
```

strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="show_map_string">Mostra la mappa</string>
    <string name="location_string">Digita l\'indirizzo</string>
</resources>
```

Во время компиляции приложения автоматически генерируется класс R.java. Java-код использует R-класс для доступа к ресурсам.

Наконец, другие файлы ресурсов могут ссылаться на строки, которые вы определили как @string/string_name. Вы также можете получить доступ к этим строкам из java-кода, но на этот раз вы делаете это так: R.string.string_name.

Другой вид ресурса – Layout или файл лейаута. Layout-

файл определяет на что будет похож пользовательский интерфейс для каждой части (каждого экрана) вашего приложения. Эти файлы так же записаны в xml, несмотря на то, что некоторые инструменты позволяют вам создавать лейаут визуально, вручную, и затем эти инструменты сгенерируют xml для вас.

К примеру, Eclipse создает Layout-файлы в каталоге res/layout вашего приложения. И вы можете получить доступ к Layout в Java как R.layout.layout_name. Так же вы можете получить доступ к Layout в других файлах ресурсов как @layout/layout_name.

Конец ознакомительного фрагмента.