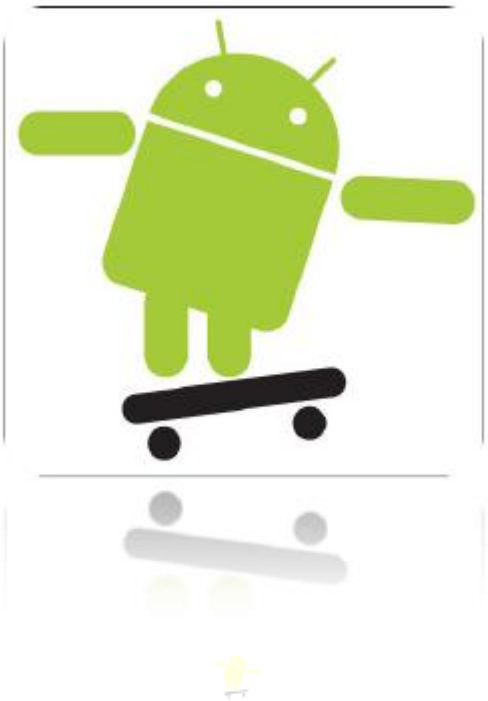
andbook!

release.002



Программирование на Android

Учебник от сообщества anddev.org.

написано Nicolas Gramlich
перевел google, ПРОМТ и Юсупов Илья

Проверь последнюю версию на http://andbook.anddev.org
А последнюю версию перевода на http://vdgames.ru

Содержание

- Содержание 2
- Если у Вас Windows 3.1 (эту главу увидят только под Windows 3.1) 3
- Предисловие/как читать этот учебник 5
- От переводчика 5
- Вступление 6
- Что такое Android - GPhone? 7
    - Android сверху 8
    - Первые недели 11
    - Dalvik.equals (Java) == false 12
- Android Developer Challenge 13
- Создание приложений Андроида 14
    - Анатомия приложения Андроида 14
    - Пользовательские интерфейсы Андроида 17
- AndroidManifest.xml 20
    - Основное 20
- Ресурсы и волшебный R.java 24
    - Ресурсы 24
    - Альтернативные Ресурсы & Локализация 26
    - Волшебный R.java 27
- Hello World - Андроид путь. 28
    - Установка Андроид SDK 29
    - Инсталляция выполнена, давайте писать код 30
    - UIs через Java 37
    - System.out.println (…)? 38
    - Использование Intent 40
    - Запуск дочерних Activity 40
    - Передача данных дочерним Activity 46
- Важные Layout и ViewGroup 49
- Библиография 49
- Lost chapters 50

powered by anddev.org

Предисловие / Как читать этот учебник

Этот документ был написан для разработчиков, которые до этого работали с Java ™, и хотят начать разработку для платформы Android. Я постарался сделать учебник как можно нагляднее, приводя примеры кодов везде, где это надо. Также я попытался вставить как можно больше иллюстраций, чтобы оживить процесс обучения и не напрягать глаза читателя.

Но, к сожалению кодирование - еще не все, что нужно, чтобы узнать о основных фактах платформы Android и в полной мере понять их. Они описаны в первых страницах. Их не обязательно читать, все эти страницы, но желательно. Можете рассматривать это, как своего рода «ссылку». Прочтя это, вы будете понимать, как идет процесс работы программы. Таким образом, вы непосредственно на начале Hello World - Android Пути.

Все коды, которые вы видите в этом документе, доступны на:

http://andbook.anddev.org/sources/

Приятного чтения!

От переводчика

1. К сожалению, в английском я не силен. Переводил в основном гугл и ПРОМТ, а я пытался придать человеческий вид результату. Но иногда это было сделать очень сложно, и тогда я пересказывал суть своими словами. Я очень надеюсь, что нигде нет искажения фактов. Но если где-то я что-то не так понял и напутал, искренне прошу прощения. Красным я обозначил те места, в переводе которых я уверен не до конца, иногда в скобках рядом писал, как было в оригинале.
2. На момент написания этого абзаца и вообще перевода текста в Андроиде я тоже не разбираюсь. Потому и перевожу. Поэтому и с ним тоже мог что-нибудь напутать. Так что настоятельно рекомендую, читая, найти исходный английский вариант и отмеченные красным места проверять там. А то мало ли что).
3. Если у Вас есть более подходящий вариант перевода какого-либо места в книге, пришлите его мне! Обязательно! Тем более, что по ссылкам ниже сделать это будет очень просто).

Обратная связь с переводчиком:

Ваше мнение о переводе можете написать сюда, исправить меня в переводе чего-либо можете здесь. Все остальное – только тут. А если хочется простоты – просто пишите на адрес C.A.Y.P.O.H@ya.ru.

Введение

Open Handset Alliance выпустил Google Android SDK 12 ноября 2007 года. Удар был невероятным, почти в каждой связанной с программированием странице новостей появился пост о выпуске SDK. Группы Google были переполнены более двумя тысячами сообщений в течение первых двух дней.

Идея платформы Android удивляла и продолжает удивлять и, конечно, привлекает все больше и больше программистов каждый день. Особенно открытой архитектурой, позволяющей изменить все, что угодно, даже стандартные приложения.

*"Android – пределом является только воображение"**

* Nicolas Gramlich – админ сайта anddev.org

Что такое Android - GPhone?

За недели и месяцы до того как Google выпустила Android SDK было много слухов о так называемом GPhone. Говорили об обеспечении гуглом бесплатного общения за счет показа контекстно-зависимой рекламы на самом устройстве.

Изображение 1. Render потенциального GPhone.

Но 5 ноября 2007 Энди Rubin* заявил: "Android [Платформа] – штука более значительная и амбициозная, чем просто телефон." Google в Open Handset Alliance (OHA) обеспечивает полный набор программного обеспечения для мобильных устройств: операционная системы, связующее ПО и ключевые мобильные приложения. То, что было выпущено вначале, не было конечным продуктом, а всего лишь "First Look SDK" (Первый взгляд на SDK), что многие не поняли. Многие новостные сайты начали ворчать, что тут уйма багов и мало документации. Но большинство говорили, что Android не багованее, чем любое другое программное обеспечение на начальном этапе.

* Энди Рубин - директор Google мобильных платформ

Android сверху

Давайте взглянем, что ОНА выделяет в своей платформе Android:

Открытость

"Android был построен с нуля до состояния, позволяющего разработчикам создавать привлекательные мобильные приложения, которые могут в полной мере воспользоваться всеми функциями телефона. Он построен по-настоящему открыто. Например, приложение может обращаться к любым из основных функций телефона, таким, как принятие вызовов, отправка текстовых сообщений или камера, позволяющим разработчикам создавать более полезные и помогающие пользователям приложения."

Это правда, как разработчик вы можете делать все, от отправки коротких сообщений с только двумя строками кода, до замены даже домашнего экрана вашего устройства. Можно легко создать и полностью настроить операционную систему в течение нескольких недель, не предоставляя больше стандартные приложения Google пользователю.

"Android построена на открытом ядра Linux. Кроме того, она использует пользовательскую виртуальную машину, которая была предназначена для оптимизации памяти и аппаратных ресурсов в мобильной среде. Android имеет открытый исходный код, в него могут быть включены новые современные технологии по мере их появления. Эта платформа будет продолжать развиваться, пока разработчики производят новые приложения".

Здесь Google говорит о так называемых Dalvik виртуальной машине (DalvikVM), которая основана на регистрах. Виртуальная машина, разработана и написана Dan Bornstein и некоторыми другими инженерами Google, чтобы быть важной частью платформы Android. В словах "на основе регистров" мы находим первое отличие от нормальных виртуальных машинах Java (JVM), которые основаны на стеке. См. "Dalvik.equals (Java) == FALSE" - главу для более подробной информации по этому вопросу.

Все приложения равны

"Android не проводит различия между основными телефонными приложениями и приложениями сторонних разработчиков. Все они построены на равном доступе к возможностям телефона, предоставляя пользователям широкий спектр приложений и услуг. С устройствами, построенными на платформе Android, пользователи будут иметь возможность в полной мере адаптировать телефон для своих интересов. Они могут поменять домашний экран телефона, стиль звонка, либо любого из приложений. Они даже могут поручить своим телефонам использовать свою любимую фотографию при просмотре приложений для обработки просмотра всех фотографий. "

Опять все это правда. Разработчики могут 100% настроить свое Android-устройство. Система связи Андроида построена на основе так называемых целей, которые являются просто строкой (возможно, с некоторыми параметрами), определяющей действия, которые должны быть обработаны. Примером этого является:

"android.provider.Telephony.SMS_RECEIVED"

Можно просто слушать о этом Intent, написав 5 строк определений. Затем до системы дойдет, что есть более чем одно приложение, которое хочет обрабатывать этот Intent, и она попросит пользователя выбрать, какое из приложений будет обрабатывать этот Intent.

Разрушение ограничений приложения

"Android разрушает барьеры к созданию новых и новаторских приложений. Например, разработчик может объединить информацию из Интернета с данными на мобильном телефоне - такими, как контакты пользователя, календарь, или географическое расположение - и обеспечить показ информации, более соответствующий пользователю. С Android, разработчик может создать приложение, которое будет позволять пользователям просматривать расположение своих друзей и предупреждать, когда они находятся в непосредственной близости, давая им возможность пообщаться".

Быстрая и простая разработка приложений

"Android предоставляет доступ к широкому спектру полезных библиотек и инструментов, которые могут быть использованы для создания сложных приложений. Например, Android дает разработчикам возможность получить расположение устройства и позволяет устройствам взаимодействовать друг с другом. Кроме того, Android содержит полный набор инструментов, которые были созданы с нуля, вместе с платформой, обеспечивающий разработчикам высокую производительность и глубокое понимание их приложения ".

После революции Web 2.0, создание наполненных информацией приложений в течение минут больше не является иллюзией. Андроид принес в разработку неизвестные скорости. Позвольте мне показать пример:

Недавно я споткнулся об умное слово ‘DrivingDirections’ в документации андроида.

Подумал-сделал.

Изображение 2. Google реализация DrivingDirections в Андроиде.

Процесс разработки приложения в изображении выше занял приблизительно полтора часа! (Включая простой пользовательский интерфейс и все изображения, которые Вы видите). Можно было создать такое сложное приложение на любой другой мобильной платформе? - Не.

Дополнительно в это приложение могло быть добавлено определение текущей позиции устройства через GPS с совсем небольшим количеством строк. Google подчеркивает мощность навигации Андроида на основе служб местоположения. Карты Google являются настолько четкими в пределах Андроида, как будто они был разработаны только для Андроида. Можно интегрировать, полностью изменяют масштаб изображения и перетаскивают карту, добавляя только три (!) символа в java-коде стандартного приложения Андроида и три строки XML-кода. Другими хорошими особенностями, которые удобны с Андроидом, являются Анимации и воспроизведение музыки. Начиная с версии m5 Андроид SDK содержит функции для прямого и обратного Геокодирования и, в дополнение к mp3, воспроизведения: ogg, MIDI и некоторых других форматов.

Первые недели

К сожалению разработчики должны были иметь дело с не полностью осуществленным preview-SDK (первая версия, билд 'm3-rc20'), где некоторые ключевые части SDK даже были плохо зарегистрированы. Была острая нехватка документации, заявление от Google:

“Если оно не было документировано - оно еще не претендует на работу. “

Многие из разработчиков не понимали, что первые выпущенные SDK были предварительными просмотрами разработчика, где должны были ожидаться изменения в API.

Другим раздражающим багом был изменчивый звук эмулятора, который, было сказано, пофиксен в билде 'm3-rc37a' приблизительно 4 недели спустя, но все еще случался на некоторых установках до m5.

Dalvik.equals (Java) == false

Почему “Dalvik”? - Виртуальную машину Dalvik назвал Bornstein в честь рыбацкой деревни Dalvik в Исландии, где некоторые из его предков жили.

Dalvik - название виртуальной машины Андроида. Это - виртуальная машина только для интерпретатора, которая выполняет файлы в Dalvik Executable (*.dex) формате, формате, который оптимизирован для эффективной памяти и распределяемого памятью выполнения. Виртуальная машина основана на регистре, и она может выполнить классы, откомпилированные языковым компилятором Java, которые были преобразованы в его родной формат с использованием включенного "dx" инструмента. VM запущена на Linux 2.6 ядре, чтобы лежать в основе функциональных возможностей (таких как организация поточной обработки и управление памятью низкого уровня). DalvikVM был также оптимизирован, чтобы работать во множественных экземплярах класса с памятью низкого уровня. Несколько VMs защищают приложение от разрушения другим разрушенным Приложением.

Отличия от обычной JavaVM

JavaVM, которого можно найти на почти всех настольном компьютере в настоящее время, является основанной на стеке Виртуальной Машиной (VM). DalvikVM, с другой стороны основанный регистр, потому что мобильные процессоры оптимизированы для основанного на регистре выполнения. Также основанные на регистре VMs позволяют более быстрое время выполнения за счет программ, которых больше после трансляции.

Android Developer Challenge

Android Developer Challenge (ADC) является идеей Google поощрить 50 самых полезных приложений для Андроид.

Изображение 3. Лого ADC.

Конечно, ADC, с его 10 миллионами долларов денежного приза, привлекал даже больше программных разработчиков, чтобы создать связку действительно полезных приложений. С другой стороны много голосов сказали, что это не было никаким хорошим выбором от Google, потому что он приведет к меньшему количеству открытого кода, так как многие боялись, что их идеи украдут, в течение первых важных месяцев после выпуска SDK. Было два запланированные ADC:

- ADC I: Представлен до 14 апреля 2008

- ADC II: Эта часть начнется после того, как первые телефонные трубки основыванные на платформе, станут доступными во второй половине 2008.

В ADC I, создатели 50 наиболее перспективных программ, отправленных до 14 апреля, получат вознаграждение 25 000 $, чтобы финансировать дальнейшую разработку. Приложения, учавствовавшие в ADC, как предполагалось, будут инновационными и продемонстрируют все возможности платформы Андроида.

Создание приложений Андроида

В этой главе мы сконцентрируемся на особенностях в создавании любого Приложения андроида.

Анатомия приложения Андроида

Есть четыре стандартных блока приложения Android:
- Activity.
- Intent Receiver.
- Service.
- Content Provider.

Не у каждого приложения должны быть все четыре блока, но Ваше приложение будет написано с их некоторой комбинацией.
Как только Вы решили, в каких компонентах Вы нуждаетесь для своего приложения, Вы должны перечислить их в файле по имени AndroidManifest.xml. Это - файл XML, где Вы объявляете компоненты своего приложения и каковы их возможности и требования. Мы скоро обсудим, за что AndroidManifest.xml ответственен.

(Это могло быть написано ОЧЕНЬ криво. Тут много текста и никаких ~~картинок~~ примеров. Рекомендую потерпеть и прочесть эту теорию, зато потом Вам будет понятней. Потом все написано гораздо глаже, не волнуйтесь)

Activity

Activity – самый распространенный из четырех стандартных блоков Андроида. Activity обычно - единственный экран в Вашем приложении. Каждый Activity осуществлен как единственный класс, который расширяет базовый класс Activity. Ваш класс отобразит пользовательский интерфейс, составленный из Views, и ответит на события. Большинство приложений состоит из множественных экранов. Например, у приложения обмена сообщениями мог бы быть один экран, который показывает список контактов, второй экран, чтобы написать сообщение выбранному контакту, и другие экраны, чтобы делать обзор старых сообщений или изменить настройку. Каждый из этих экранов был бы осуществлен как Activity. Перемещение в другой экран достигнуто стартом нового Activity. В некоторых случаях Activity может возвратить значение предыдущего Activity - например Activity, которая позволяет пользователю выбирать фотографию, возвратил бы выбранную фотографию вызывающей программе. Когда новый экран открывается, предыдущий экран приостановлен и помещен на стек хронологии. Пользователь может переместиться назад через ранее открытые экраны в хронологии. Экраны могут также хотеть быть удаленными от стека хронологии, когда было бы неуместно для них остаться. Андроид сохраняет стеки хронологии для каждого приложения, начатого от начала экрана.

Intent и фильтры Intent

Андроид использует специальный класс под названием Intent, чтобы двигаться от экрана к экрану. Intent описывает то, что приложение собирается сделать. Две самых важных части структуры Intent - действие и данные к действию. Типичные значения для действия – MAIN (главный экран

приложения), VIEW, PICK, EDIT, и т.д. Данные выражены как Uniform Resource Indicator (URI). Например, чтобы рассмотреть вебсайт в браузере, Вы создали бы Intent с действием VIEW и набором данных – адресом сайта.

```
new Intent(android.content.Intent.VIEW_ACTION, ContentURI.create("http://anddev.org"));
```

Есть связанный класс, названный IntentFilter. В то время как Intent - запрос сделать кое-что, IntentFilter - описание того, что Intent Activity (или intent receiver, см. ниже), способен к обработке. Activity, который в состоянии отобразить информацию для человека, издала бы IntentFilter, который сказал, что знает, как обработать VIEW действия. Activity издает свой IntentFilters в файле AndroidManifest.xml.

Навигация от экрана к экрану достигнута достигается с помощью Intent. Чтобы переместиться вперед, Activity вызывает startActivity (myIntent). Система тогда смотрит на IntentFilter для всех установленных приложений и выбирает Activity, Intent которого фильтрует myIntent. Новому Activity сообщают о Intent, которое заставляет его начаться. Процесс решения Intent происходит, когда startActivity вызывают. Процесс предлагает две ключевых льготы:
- Действия могут многократно использовать функциональные возможности от других компонентов, просто делая запрос в форме Intent.
- Действия могут быть заменены в любое время новым Activity с эквивалентным IntentFilter.

Intent Receiver

Вы можете использовать IntentReceiver, когда Вы хотите, чтобы код в своем приложении выполнился в реакции на внешнее событие, например, когда телефон звонит, или когда сеть передачи данных доступна, или когда это - полночь. Intent Receiver не отображают UI, хотя они могут отобразить Уведомления, чтобы привести пользователя в готовность, если кое-что интересное случилось. Поглощенные получатели также регистрированы в AndroidManifest.xml, но Вы можете также регистрировать их в коде, используя Context.registerReceiver(). Ваше приложение не должно работать для его Intent Receiver, которые вызываются; система запустит Ваше приложение, в случае необходимости, когда Intent Receiver будет вызван. Приложения могут также послать свои собственные Intent Receiver другим с Context.broadcastIntent().

Service

Service - код, который долговечен и выполняется без UI. Хороший пример этого - универсальный проигрыватель, запускющий песни из плейлиста. В приложении универсального проигрывателя, вероятно, были бы одно или более Activity, которые позволяют пользователю выбирать песни и запускать их. Однако, воспроизведение самой музыки не должно быть обработано Activity, потому что пользователь будет ожидать, что музыка продолжит играть даже после сворачивания проигрывателя. В этом случае, деятельность универсального проигрывателя могла запустить Service, используя Context.startService(), чтобы работать на заднем плане и сохранить воспроизведение музыки. Тогда система сохранит воспроизведение музыки, пока оно не закроется само. (Вы можете узнать больше о приоритете, данном службам в системе, читая Цикл

Жизни Приложения Андроида). Отметьте, что Вы можете соединиться с Service (и запустить его, если он уже не работает) с методом Context.bindService(). Когда есть подключение с Service, Вы можете общаться с этим через интерфейс, выставленный Service. Для Service музыки это могло бы позволить Вам приостанавливать, перематывать, и т.д.

Content Provider

Приложения могут хранить свои данные в файлах, базе данных SQLite, персональных настройках или любом другом механизме, который имеет смысл. Content Provider, однако, полезен, если Вы

хотите, чтобы данные Вашего приложения были разделены с другими приложениями. Content Provider - класс, который осуществляет стандартный набор методов, чтобы позволить другим приложениям сохранять и восстанавливать тип данных, которые обработаны другим(that) Content Provider.

Пользовательские интерфейсы Андроида

Пользовательские интерфейсы (UI) в Андроиде могут быть созданы двумя путями, через XML-код или в java-коде. Создание структуры графического интерфейса пользователя в XML очень предпочтительно, потому что по принципу Образцового управления средства просмотра, UI должен всегда отделяться от логики программы. К тому же, приспосабливание программы от одной разрешающей способности экрана до другой намного более просто. Определение UI в XML очень похоже к создаванию общего документа HTML, где Вы имеете то есть такой простой файл:

```
<html>
     <head>
          <title>Page Title</title>
     </head>
     <body>
          The content of the body element.
     </body>
</html>
```

Все равно как в Андроидовских XML-Layouts. Все хорошо структурировано и может быть выражено древовидными структурами:

```
<?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="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello World"/>
</LinearLayout>
```

Иерархия Элементов Экрана

Основной функциональный модуль приложения Android - Activity - объект класса android.app.activity. Activity может сделать много вещей, но отдельно у него нет присутствия на экране. Чтобы дать Вашему Activity присутствие на экрана и проектировать его UI, Вы работаете с Views и Viewgroups - основными единицами выражения пользовательского интерфейса на платформе Андроида.

Views

View - объект, расширяющий базовый класс android.view.view . Это - структура данных, свойства которой сохраняют Layouts и информационное наполнение для определенной прямоугольной области экрана. Объект View обрабатывает измерение, его схему размещения, рисунок, изменения центра, прокрутку, и клавиши/знаки для области экрана, которую он представляет. Класс View служит базовым классом для всех графических фрагментов - ряд полностью

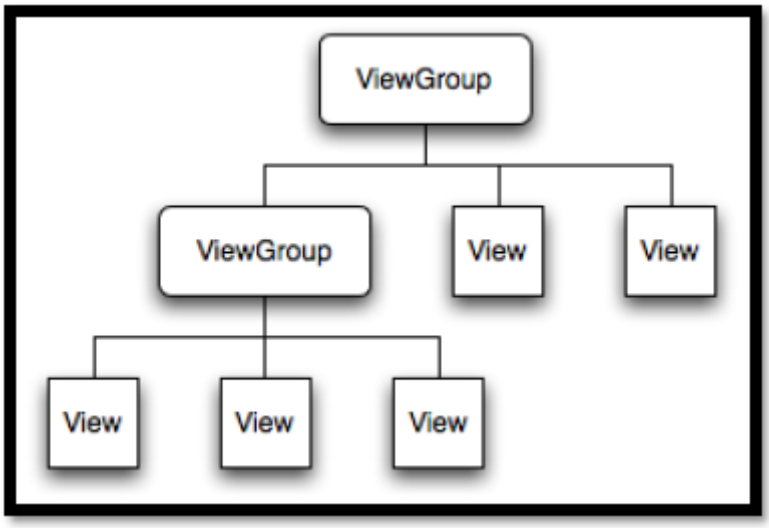
осуществленных подклассов, которые рисуют интерактивные элементы экрана. Графические фрагменты обрабатывают свое собственное измерение и рисунок, таким образом Вы можете использовать их, чтобы создать Ваш UI более быстро. Список доступных графических фрагментов включает TextView, EditText, Button, RadioButton, Checkbox, ScrollView и т.д.

Viewgroups

Viewgroup - объект класса android.view.viewgroup. Viewgroup - специальный тип объекта View, функция которого - содержать набором View и Viewgroup и управлять ими. Viewgroups позволяют Вам добавлять структуру к Вашему UI и создавать сложные элементы экрана, к которым можно обратиться как к единственному объекту. Класс Viewgroup служит базовым классом для Layouts - ряда полностью осуществленных подклассов, обеспечивающего общие типы Layouts экрана. Layouts дают Вам способ встроить структуру для ряда View.

UI с древовидной структурой

На платформе Андроида Вы определяете UI Activity использование дерева View и Viewgroup узлов, как показано в диаграмме ниже. Дерево может быть столь же простым или сложным, как Вы его сделаете, и Вы можете построить его, используя наборы предопределенных графических фрагментов и Layouts Андроида, или заказных типов View, которые Вы создаете самостоятельно.



Изображение 4. UI Андроида – древовидная структура.

Чтобы прикрепить дерево к экрану и отрендрить его, Ваш Activity вызывает свой метод setContentView() и передает информацию на корневой объект узла. Как только у система Андроида получает информацию на корневой объект узла, она начинает работать непосредственно с узлом, чтобы измерить, и отрисовать дерево. Когда Ваш Activity становится активным и получает приоритет, система регистрирует Ваш Activity и просит корневой узел измерить и отрисовать дерево. Тогда корневой узел просит, чтобы его дочерние вершины отрисовали себя - в свою очередь, каждый Viewgroup узел в дереве ответственен за отрисовку его

прямых дочерних узлов. Как упомянуто ранее, у каждой группы View есть ответственность измерения ее доступного пространства, расположения ее дочерних узлов, и вызов draw() на каждом дочернем узле, чтобы позволить все им рендрить себя. Дочерние узлы могут просить размер и местоположение в родителе, но у родительского объекта есть конечное решение, где и насколько большой каждый ребенок может быть.

Сравнение Андроида Элементы UI к Swing Элементы UI

Поскольку некоторые разработчики, которые читают это, возможно, нашли, что UIs схож с Swing, сейчас будет немного общих черт между Андроидом и Swing.
- Activity в Андроиде - почти (J) Frame в Swing.
- View в Андроиде - (J) Component в Swing.
- TextViews в Андроиде - (J) TextField в Swing.
- EditTexts в Андроиде - (J) TextField в Swing.
- Button в Андроиде - (J) Button в Swing.

Установка слушателей к View в Андроиде является почти тем же самым, чем и в Swing.

```
// Android
myView.setOnClickListener(new OnClickListener(){ ...

// Swing
myButton.addActionListener(new ActionListener() {...
```

AndroidManifest.xml

AndroidManifest.xml - необходимый файл для каждого приложения Android. Он расположен в папке приложения и описывает глобальные значения для Вашего пакета, включая прикладные компоненты (действия, службы, и т.д), который пакет выставляет 'внешнему миру', какие данные каждое из Activity приложения может обработать, и как они могут быть начаты.

Важная вещь к упоминанию об этом файле - свой так называемый IntentFilters. Эти фильтры описывают где и когда Activity может быть начат. Когда Activity (или операционная система) хочет выполнить действие, такое, как открыть Web-страницу или открыть экран выбора, это создает объект Intent. Этот Intent может хранить информацию, описывающую, что Вы хотите сделать, какие данные необходимы, чтобы достигнуть этого и другую информацию. Андроид сравнивает информацию в объекте Intent с IntentFilters, выставленным каждым приложением, и находит Activity, соответствующие, чтобы обработать данные или действия, определенные вызывающей программой. Если есть больше чем одно приложение, способное к обработке этого Intent , пользователя спрашивают, каким приложением он предпочел бы обрабатывать это.

Помимо объявления Activity, Content Provider, Service и Intent Receivers Вашего приложения, Вы можете также определить разрешения в AndroidManifest.xml.

Основное

Очень простой AndroidManifest.xml выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="org.anddev.android.hello_android">
      <application android:icon="@drawable/icon">
            <activity android:name=".Hello_Android"
                        android:label="@string/app_name">
                  <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.LAUNCHER" />
                  </intent-filter>
            </activity>
      </application>
</manifest>
```

- Почти каждый AndroidManifest.xml (так же как многие другой Андроид файлы XML) будет включать объявление пространства имен в его первый элемент. Это делает множество стандартных атрибутов Андроида доступным в файле, который будет использоваться, чтобы снабдить большинством данных для элементов в том файле.
- Почти каждый AndroidManifest.xml включает единственный <application> тэг, который непосредственно содержит несколько тэгов, описывающих Applications, IntentReceivers, и т.д., которые доступны в этом приложении.
- Если Вы хотите сделать Activity запускаемым непосредственно через пользователя, то Вы должны будете прописать ему поддержку действия MAIN и категории LAUNCHER. Результат как показано здесь:

Picture 5. Напрямую запускаемый Activity.

Дальше следует, детальный разбор структуры файла AndroidManifest с описыванием всех доступных <tags>, с примером для каждого. (Примеры не везде, но я точно ничего не потерял. И фраза “для каждого” в англ. версии (“for each”) тоже была. Не знаю, где они потеряли эти примеры. – прим. переводчика). Кто чей потомок, можно определить по отступу:

<manifest>

Это корневой узел каждого `AndroisManifest.xml`. Он содержит пакета атрибутов, который указывает на какой-то конкретный пакет в Activity (Получилось не совсем по-русски, но, вообщем, фишка в том, что в коде есть пакет Activity, а вот параметры этого пакета описаны в этом теге. - прим. переводчика). Другой путь к Activities будет базироваться относительно его значения.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="org.anddev.android.smstretcher">
```

<uses-permission>

Описывает разрешения безопасности, которы нужно предоставить Вашему пакету. Количество не ограничено.

```
<uses-permission android:name = "android.permission. RECEIVE_SMS"/>
```

<permission>

Объявляет разрешение безопасности, которое может использоваться, чтобы указать, какие приложения могут обратиться к компонентам или особенностям в этом пакете. Количество не ограничено.

<instrumentation>

Объявляет код контрольно-измерительного компонента, который доступен, чтобы проверить функциональные возможности этого или другого пакета. См. Оснащение аппаратурой для большего количества подробностей. Количество не ограничено.

<application>

Корневой элемент, содержащий объявления компонентов на уровне приложения, содержится в пакете. Этот элемент может также включать глобальные и/или заданные по

умолчанию атрибуты для приложения, такие как метка, значок, тема, требование разрешения, и т.д. Количество – от нуля до единицы.

```
<application android:icon="@drawable/icon">
```

<activity>

Activity - первичная вещь для приложения, чтобы взаимодействовать с пользователем. Экраном, который пользователь видит, запуская приложение, является Activity, и большинство других экранов, которые они используют, будет осуществлено как отдельные действия, объявленные с дополнительными тэгами Activity.

```
<activity android:name=".Welcome" android:label="@string/app_name">
```

Примечание: *у Каждой Activity должен быть <activity> тэг в manifest, выставлено ли это миру или предназначено для использования только в пределах собственного пакета. Если у Activity не будет никакого тэга соответствия в manifest, то Вы не сможете использовать ее.*

<intent-filter>

Объявляет, какие Intent компонент поддерживает. В дополнение к различным видам значений, которые могут быть определены под этим элементом, атрибуты могут быть даны здесь, чтобы поставлять уникальную метку, значок, и другую информацию для описываемого действия.

```
<intent-filter>
```

<action>

Тип действия, который компонент поддерживает. Пример:

```
<action android:name="android.intent.action.MAIN" />
```

<category>

Тип категории, который компонент поддерживает. Пример:

```
<action android:name="android.intent.category.LAUNCHER" />
```

<data>

Тип MIME, URI схема или URI путь поддержки компонента.

Вы можете также связаться 1+ части метаданных с Вашим Activity:

<meta-data>

Добавляет новую часть метаданных к Activity, которую клиенты могут восстановить через ComponentInfo.metaData.

<receiver>

IntentReceiver позволяет приложению сообщать о замене данных или о действиях, которые происходят, даже если приложение не выполняется в настоящее время. Как и с тэгом activity, в этот тег Вы можете произвольно включить больше одного <intent-filter> или <meta-data>, так же, как с <activity>.

```
<receiver android:name=".SMSReceiver">
```

<service>

Service - компонент, который работает на заднем плане произвольное количество времени. В этот тег, как с тэгом activity, Вы можете произвольно включить один или более <intent-filter> или <meta-data>.

<provider>

ContentProvider - компонент, который управляет постоянными (persistent) данными и открывает к ним доступ другим приложениям. Вы можете также произвольно присоединить один или более <meta-data>, как и в activity. <intent-filter> тут нету.

Конечно, все <теги> должны быть или так </закрыты>, или так, <напрямую/>.

Ресурсы и волшебный R.java

Ресурсы проекта и R.java очень близко связаны.

Ресурсы

Ресурсы - внешние файлы (не код), которые используются Вашим кодом, закомпилированы в Ваше приложение и встраиваются в него во время работы. Андроид поддерживает многие различные виды файлов ресурсов, включая XML, PNG и JPEG. Файлы XML имеют сильно различающиеся форматы в зависимости от того, что они описывают. Ресурсы описаны в исходном коде, и файлы XML откомпилированы в двоичный код для быстрой и эффективной загрузки. Строки сжаты в более форму, более экономящую память.

Список ресурсов

Типы ресурсов и их местоположение:

- layout-файлы - “/res/layout/” .
- изображения - “/res/drawable/” .
- анимация - “/res/anim/” .
- стили, строки и массивы - “/res/values/” .
    - *Названия не могут отличаться:*
    - ‘arrays.xml’ для определения массивов.
    - ‘colors.xml’ для определения цветов.
        - #RGB, #ARGB, #RRGGBB, #AARRGGBB
    - ‘dimens.xml’ для определения размеров (dimensions).
    - ‘strings.xml’ для определения строк.
    - ‘styles.xml’ для определения стилей объектов.
- необработанные файлы вроде mp3 или видео - “/res/raw/”

Использование ресурсов в коде

Для использования ресурса в коде нужно знать только полный ID ресурса и в какой тип объекта Ваш ресурс был откомпилирован. Вот синтаксис обращения к ресурсу:

R.resource_type.resource_namет или android.R.resource_type.resource_name

resource_type - подкласс R, который содержит определенный тип ресурса. resource_name - атрибут ресурсов, определенный в файлах XML, или имя файла (без расширения) для ресурса, определенных другими типами файла. Каждый тип ресурса будет добавлен в подкласс R, в зависимости от его типа.

Ресурсы, откомпилированные Вашим приложением, могут быть использованы без названия пакета (просто как R.resource_type.resource_name). Андроид содержит многие стандартные ресурсы, такие как стили экрана и фоны кнопки. Обращаться к ним в коде Вы можете через android.R.resource_type.resource_name, для примера: android.R.drawable.button_background.

Ссылка на Ресурсы

Значение в атрибуте или ресурсе может также быть ссылкой на другой ресурс. Это часто используется в layout файлах, чтобы хранить строки (таким образом можно локализовать приложение) и изображения (находящиеся в другом файле), хотя ссылка может быть на любой тип ресурса, включая цвета и числа.

Например, если у нас есть ресурсы с цветами, мы можем написать layout файл, который установит цвет текста на значение, содержащееся в одном из ресурсов:

```
<EditText
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@color/opaque_red"
      android:text="Hello, World!" />
```

Обратите внимание на префикс "@", показывающий, что это - ссылка на ресурс, текст после него - название ресурса в форме @[пакет:]тип/имя. В примере мы не определяем пакет, потому что ссылаемся на ресурс в нашем собственном пакете. Чтобы сослаться на системный ресурс, Вы должны были бы написать:

```
<EditText
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@android:color/opaque_red"
      android:text="Hello, World!" />
```

В следующем примере, мы используем ссылку на ресурс, храня строки в layout файле так, чтобы они могли быть локализованы:

```
<EditText
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@android:color/opaque_red"
      android:text="@string/hello_world" />
```

Альтернативные Ресурсы & Локализация

Альтернативные Ресурсы и Локализация - серьезная проблема, достаточно хорошо решенная в Андроиде. Обычно чаще всего Вы должны были бы проектировать UI, хорошо подходящий для каждого из возможных разрешений экрана одновременно, что почти невозможно.

Вы можете добавлять в свое приложение различные UI, языки или поддержку устройств с различной конфигурацией харда.

Заметьте, что даже если Вы добавите много разных языков, UI и всех других ресурсов, SDK сам определит набор ресурсов, который будет использоваться. К примеру, Андроид сам догадается, где какой язык Вам нужен и выберет его. Или UI. Чтобы включать дополнительные ресурсы, создайте параллельные папки с ресурсами и к каждому названию через черточку добавьте параметр (спецификатор), куда эта папка относится (язык, ориентация экрана, точки на дюйм, разрешение, …). Вот, например, у этого проекта есть английская и немецкая локализация (здесь только строки):

```
MyApp/
      res/
            values-en/
                  strings.xml
            values-de/
                  strings.xml
```

Андроид поддерживает несколько типов спецификаторов, с различными значениями для каждого. Добавьте их концу названия папки ресурса, отделив от названия черточкой. Вы можете добавить много спецификаторов, отделяя их друг от друга черточками. Например, папка, содержащая drawable ресурсы только для определенной конфигурации:

```
MyApp/
      res/
            drawable-en-rUS-port-92dpi-finger-keyshidden-12key-dpad-480x320/
```

Более того, Вы можете определить только несколько определенных опций конфигурации, для которых определен ресурс.

```
MyApp/
      res/
            drawable-en-rUS-finger/
            drawable-port/
            drawable-port-160dpi/
            drawable-qwerty/
```

Андроид выберет, какой из различных основных файлов ресурса подходит лучше всего во время выполнения, в зависимости от текущей конфигурации устройства.

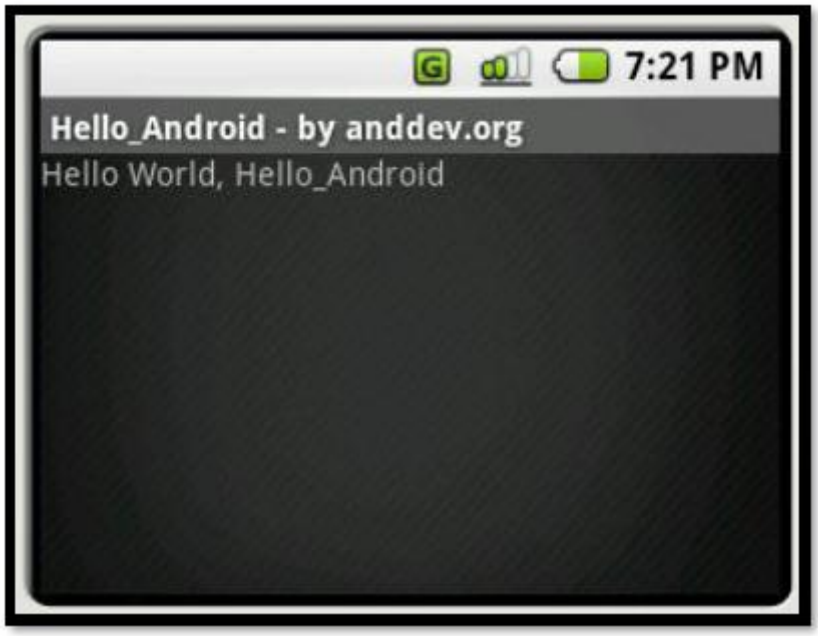
Волшебный R.java

R.java проекта - автоматически сгенерированный файл, индексирующий все ресурсы Вашего проекта. Вы используете этот класс в своем исходном тексте как своего рода способ обратиться к ресурсам, которые Вы включили в свой проект. Это особенно важно, учитывая особенности интегрированных сред разработки, потому что позволяет Вам быстро и в интерактивном режиме определять местонахождение определенной информации, которую Вы ищете. Дополнительно во время компиляции Вы получаете уверенность, что ресурс, который Вы хотите использовать, действительно существует.

Hello World - Андроид путь.

В этой самой первой обучающей программе каждый узнает, как создать Приложение на Андроиде, используя XML layout. Имейте в виду, что, использование XML layout предпочтительно!

На что это будет наконец похоже:



Изображение 6. Первое приложение на Андроиде – финальная версия (версия SDK – m5).

Но сначала нужно загрузить и поставить Android SDK…

Установка Андроид SDK

Последний Андроид SDK для Windows, Mac и Linux может всегда найти тут:

http://code.google.com/android/download.html

Загрузите его и распакуйте в облюбованное Вами местечко для SDK на Вашем HDD.

Андроидовские Инструменты Разработки (ADT)

Андроид предоставляет аддон к Eclipse под названием 'ADT', чтобы сделать программирование и отладку проще. ADT обеспечивает свободный доступ к LogCat, Android-Manifest/Resource-Editor, файлам, потокам, управлению памятью, позволяет симулировать входящий звонок/смс, и т.д …

Установка плагина ADT для Eclipse

Чтобы загрузить и установить дополнение к программе ADT, следуйте за шагами, которые Google предоставляет разработчикам:

1. Запустите Eclipse, затем выберите **Help > Software Updates > Find and Install....**
2. В диалоге, который появляется, выберите **Search for new features to install** и нажать **Next**.
3. Нажмите **New Remote Site**.
4. В появившемя диалоговом окне, введите имя для удаленного сайта (например, Аддон к Андроида) и URL (https://dl-ssl.google.com/android/eclipse/). Нажмите **OK**.
5. Теперь Вы должны видеть новый сайт, проверенный и добавленный к списку поиска. Нажмите **Finish**.
6. В следующем диалоговом окне **Search Results**, выберите for **Android Plugin** > **Developer Tools**. Программа проверит "**Android Developer Tools**", и "**Android Editors**". **Android Editors** - дополнительно, но это рекомендуется. Если вы хотите установить что-то, нужен плагин WST, о нем на этой странице написано выше. Нажмите **Next**.
7. Прочтите лицензионное соглашение и выберите **Accept terms of the license agreement**, если согласны. Нажмите **Next**.
8. Нажмите **Finish**.
9. Плагин ADT не подписан; Вы можете принять инсталляцию так или иначе, нажав **Install All**.
10. Перезапустите Eclipse.
11. После рестарта, обновите свои персональные настройки Eclipse**,** чтобы указать на каталог SDK:

a. Выберите **Window** > **Preferences...,** чтобы открыть панель Персональных настроек. (Mac OS X: **Eclipse** > **Preferences**)

b. Выберите **Android на** левой панели.

c. Для Местоположения SDK в основной панели, нажмите **Browse...** и определите местонахождение каталога SDK.

d. Нажмите **Apply**, потом **OK**.

Обновление плагина ADT

После установки плагина ADT его следует обновить:

1. Выберите **Help > Software Updates > Find and Install....**
2. Выберите **Search for updates of the currently installed features** и нажмите **Finish**.
3. Если какое-нибудь обновление для ADT доступно, выберите его и установите.

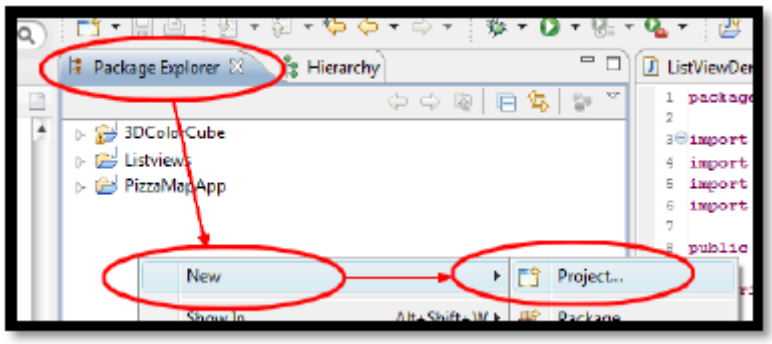
Инсталляция выполнена, давайте писать код

Перед началом кодирования мы должны создать новый Проект Андроида.

Создание нового Проекта Андроида

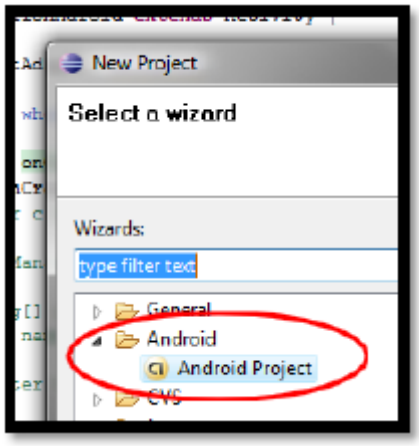
Чтобы создать новый проект, просто откройте **Package Exlporer** в Eclipse, щелкнув правой кнопкой мыши где-нибудь на пустом месте и выберите:

New > Project…



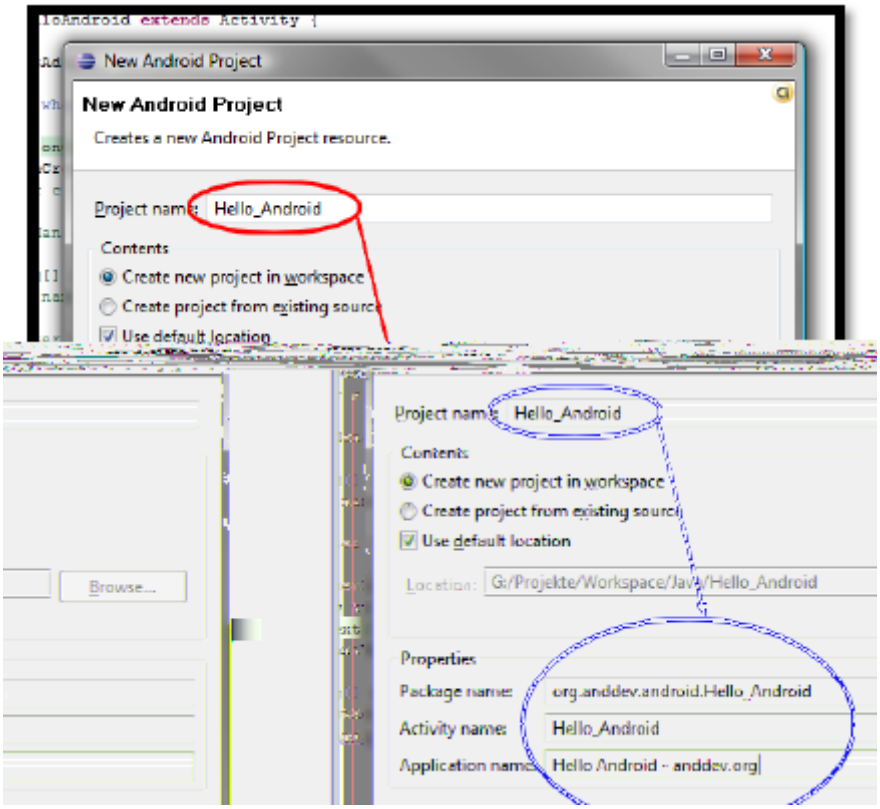
Изображение 7. Первое Приложение к Андроиду - Шаг 1 .

Выберите **Android > Android Project**



Изображение 8. Первое Приложение к Андроиду - Шаг 2 .

Заполните форму значениями, соответствующими цели Вашего приложения...



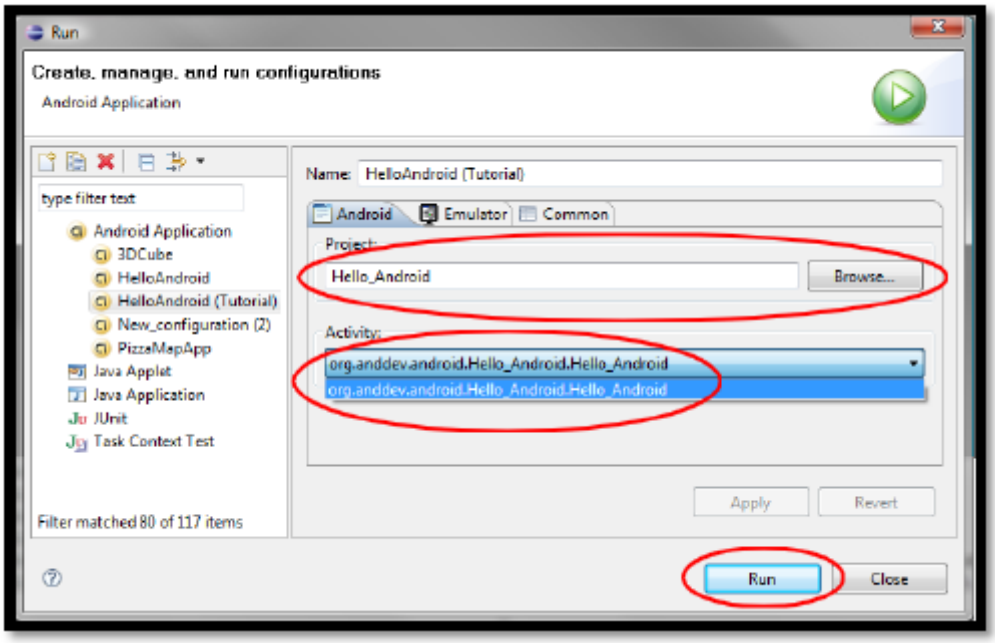
Изображение 9. Первое Приложение к Андроиду - Шаг 3 .

4. Это все - файлы для Вашего первого Приложения к Андроиду (не паникуйте, почти все они - файлы ресурсов).

Изображение 10. Первое Приложение к Андроиду - Шаг 4

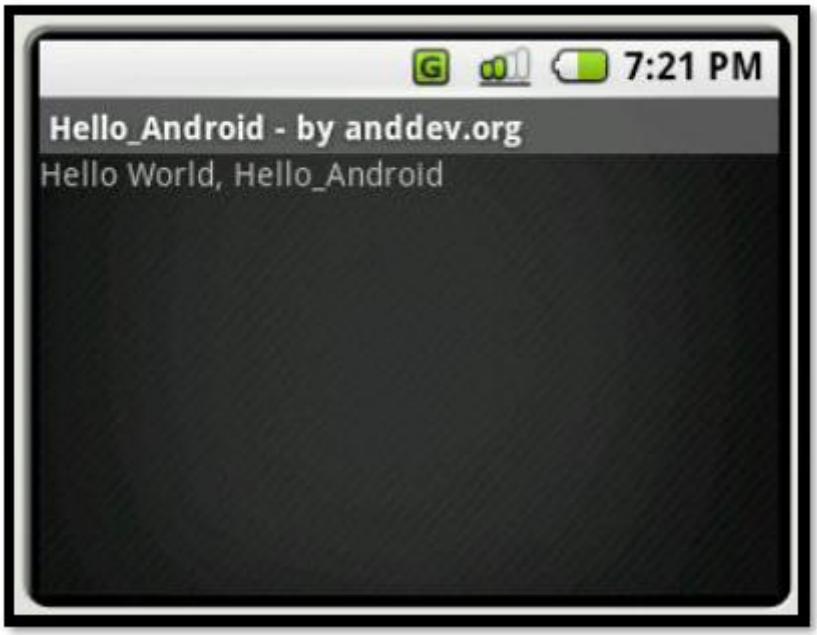
Ха, что все эти файлы тут делают?

Поскольку Вы теперь создали свой первый Проект Андроида, Вы видите, что в новом проекте появляется вот такой набор файлов.



Изображение 12. Создание конфигурации запуска.

Теперь Вы можете запустить свое приложение, нажав . Запустив его, Вы увидите это:



Изображение 13. Первое приложение на Андроиде – результат (версия SDK – m5).

После того, как приложение откроется, Вы сможете также увидеть, что Ваше приложение появилось в быстром меню, отображенне с заданным по умолчанию значком:

Изображение 14. Первое приложение на Андроиде – иконка в быстром меню (версия SDK – m5).

UIs через Java

То, что мы только что сделали через XML-код, может быть достигнута через взлом некоторых строк java кода. Помните, как мы устанавливали свой main.xml как ContentView в xml-примере. Мы достигали этого, делая следующее:

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

Activity.setContentView(…) использует View как параметр. Мы используем это, установив простой TextView как наш ContentView.

```
package org.anddev.android.Hello_Android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Hello_Android extends Activity {
      /** <uau\Z_lky  dh]^Z Activity kha^Zg */
      @Override
      public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
            /* Fu ohlbf m\b^_lv hq_gv ijhklhc, l_dkl
            * ^ey wlh]h fu kha^Zg_f\uc h[t_dl TextView. */
            TextView tv = new TextView(this);
            /* Ibr_f \ \gh\v kha^Zgguc l_dkl wlm k/jhdm
            tv.setText("Hello Android - by: anddev.org \n" +
                  "This is soooo simple =D ");
            /* =h\hjbf gZr_fm Activity ihdZaZlv l_dk/
            this.setContentView(tv);
      }
}
```

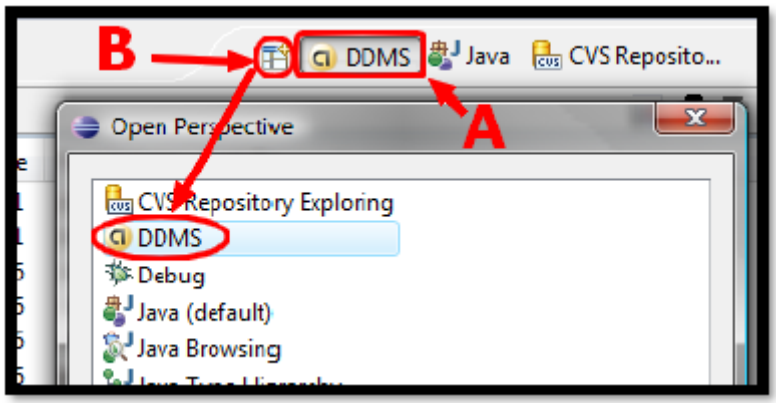
System.out.println (…)?

Отладка в Андроиде не может быть сделана, используя System.out.println (…), потому что, поскольку мы знаем, Андроид работает не на нормальной VM, а на DalvikVM. (Честно говоря, это сделать это можно, но маразм там еще тот, нафига оно Вам?). Но не волнуйтесь, у Андроида есть вариант покруче- LogCat.

LogCat

LogCat - часть DDMS (Сервис Отладчика Dalvik), который обеспечивает механизм для сбора отладочной информации и ее вывода. Логи различных приложений и частей системы собраны в LogCat, который может быть рассмотрен и фильтрован.

Если ты не видишь A, делай через B.



Изображение 15. Как увидеть DDMS.

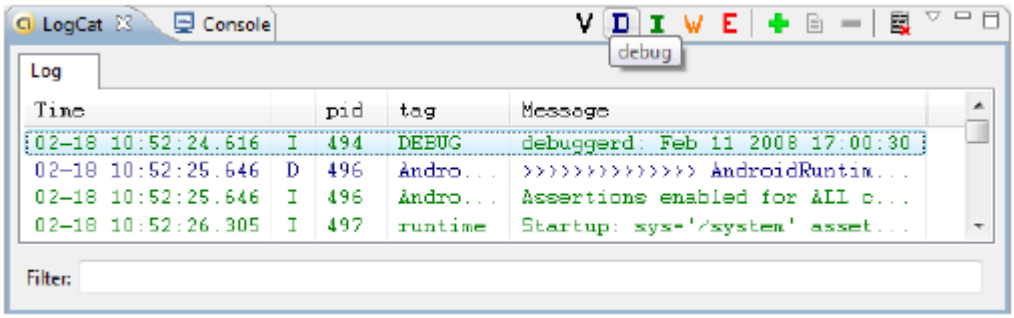
Использование LogCat

У Каждого лог-сообщения Андроида есть тэг и приоритет, связанный с ним.

Тэг лог-сообщения - короткая строка, указывающая системный компонент, из которого сообщение происходит (например, "View" для системы View). Приоритет - одно из следующих символьных значений, заказанных от самого низкого до самого высокого приоритета:

- V - Подробный (самый низкий приоритет)
- D - Отладка
- I - Информация
- W - Предупреждение
- E - Ошибка (самый высокий приоритет)

Так как мы используем Eclipse , мы можем фильтровать приоритет, нажимая кнопки *"V-D-I-W-E"* при просмотре LogCat, которые можно также видеть ниже. Вы полюбите аддон ADT хотя бы потому, что в куче сообщений без фильтров найти что-нибудь очень тяжело.



Изображение 16. LogCat.

Использовать LogCat в коде очень просто, достаточно подключить библиотеку:

```
import android.util.Log;
```

Для вывода Вашего сообщения в лог достаточно написать:

```
Log.d(" L _ ]  ^ _ [ Z k h h [ s _ g b y " K Z f h  ^ _ [ Z k h h [ s _ g b y ;
```

Если есть какой-то критический участок кода с вероятностью ошибки, можно передать сообщение в лог таким образом:

```
try{
      throw new Exception();
}catch(Exception e){
      Log.e(" L _ ]  ^ _ [ Z k h h [ s _ g b y " I j h b a h r e Z r b [ d Z, e);
}
```

Использование Intent

Как мы знаем, выполнение действий (things) выражается через Intent. Intent могут использоваться для запуска других Activity.

Запуск дочерних Activity

Одна из основных задач в приложении, которое сложнее, чем приложение HelloWorld - запустить другие Activity, особенно SubActivity. Давайте поставим такую задачу:

Мы хотим иметь что-то вроде диалога для ввода, куда пользователь будет вводить то ключевое слово, которое он хочет найти в гугле.

Создайте новый Андроидский Проект, как мы уже сделали это прежде. Первая вещь, которую надо сделать - добавить второй Activity, который мы назовем “MySecondActivity”.

В начале код будет выглядеть так:

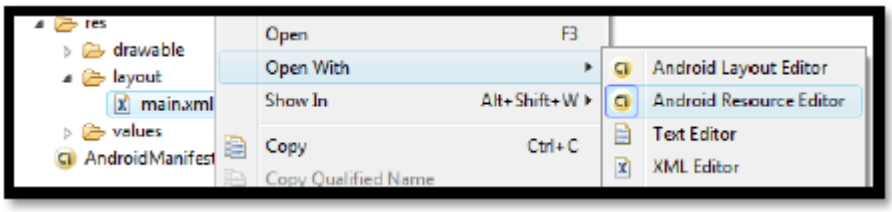
```
package org.anddev.andbook.startingsubactivities;

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

public class MySecondActivity extends Activity {
	/** <uau\Z_lky  dhjA^Zivity  kha^Zg*/
	@Override
	public void onCreate(Bundle icicle) {
		super.onCreate(icicle);
		setContentView(R.layout.main);
	}
}
```

Теперь мы добавим кнопку к первому Activity. Конечно, мы сделаем это через main.xml, а не через java UI.

Пройдите на “/res/layout/main.xml” и вы увидите код вроде этого:

```
<?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="Hello World, StartingSubactivities" />
</LinearLayout>
```

Примечание: *возможно Eclipse, открыл main.xml с его заданным по умолчанию Xml-Редактором, который не подходит для нас. Начиная с SDK-версии m5 плагин ADT включает Редактор Ресурсов с Выделением Синтаксиса.*



Изображение 17. Открытие Редактора Ресурсов.

Если бы мы запустили приложение сейчас, оно было бы похоже на:

Изображение 18. Стандартный layout (версия SDK – m5).

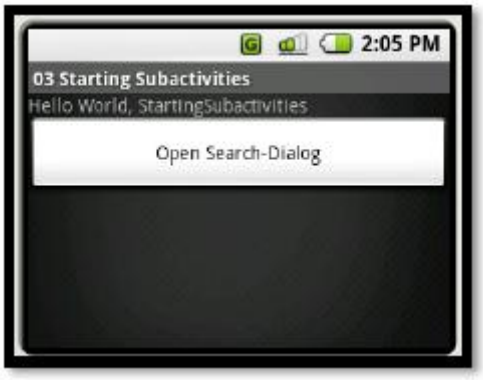
Так как мы хотим добавить кнопку к нашему приложению, мы напишем xml-код:

```
<Button
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Open Search-Dialog" />
```

Как можно увидеть, fill_parent растягивает кнопку на ширину родительского View, а wrap_content устанавливает высоту кнопки, какая требуется для наиболее корректного вида кнопки.

Таким образом, наша кнопка заполнила весь экран по ширине, а написано на ней "Open Search-Dialog", как и требовалось.

Сейчас наше приложение выглядит так:



Изображение 19. Layout с кнопкой (версия SDK – m5).

Когда теперь мы нажмем кнопку, понятное дело, ничего не случится – чтобы что-то произошло, нам нужно применить Listener, определив его на OnClickListener, с целью обработать щелчок по кнопке.

Подождите… но как мы обратимся к этой кнопке через java-код?

Нахождение View, определенных через XML, в java-коде.

Для нахождения View в java-коде, нам нужно сперва добавить к этому View атибут android:id. В этом приложении мы дадим ему id “btn_open_search”.

```
<Button
      android:id="@+id/btn_open_search"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Open Search-Dialog" />
```

Нахождение View в коде, зная его id – ничего сложного. Достаточно воспользоваться функцией findViewById(int resourceID), передав ей в качестве параметра id искомого View, который мы определили в XML.

То же самое мы можем сделать со всеми остальными объектами View: с EditText, TextView, MapView и т.д.

Примечание: *использование findViewById возможно только в тех Layout, которые были загружены в Activity через setContentView(…)!*

Вернемся к нашему примеру и добавим следующий код в функцию OnCreate(…) после setContentView(…):

```
/* G Z c ^ _ f  d g h i d m  b a  g Z XML-layout. */
Button b = (Button)this.findViewById(R.id.btn_open_search);
```

Если Eclipse не может найти класс Button, нажми Ctrl+Shift+O и в код автоматически добавится (в этом примере):

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

Обработка щелчка по виду

Мы помним, что хотели сделать нашу кнопку нажимающейся. Для этого мы просто установим анонимный OnClickListener нашей кнопке:

```
b.setOnClickListener(new OnClickListener(){
      public void onClick(View arg0) {
            // Place code to handle Button-Click here.
      }
});
```

Примечание: *Eclipse может не распознать следующий импорт отдельно:*

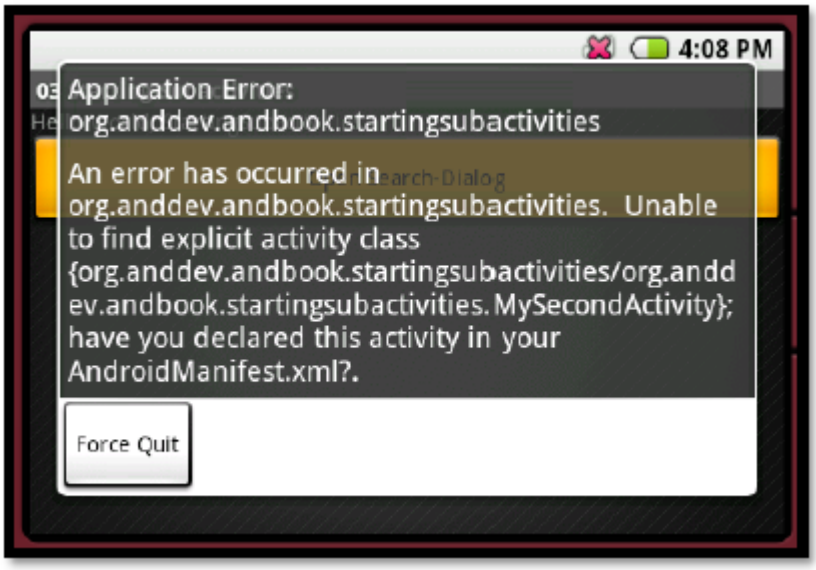
```
import android.view.View.OnClickListener;
```

Сейчас мы добавим Intent на событие щелчка для запуска нашего дочернего Activity:

```
/* Kha^Zgb Intent ^ey klZjlZ
 * MySecondActivity. */
Intent i = new Intent(
      StartingSubactivities.this,
      MySecondActivity.class);
/* HlijZ\ey_f Intent HK  qlh[u
 * kdZaZlv  qlh fu ohlbf aZbihwkl
 * MySecondActivity dZd ^hq_jg_Activity. */
startSubActivity(i, 0x1337);
```

Второй параметр startSubActivity может быть уникальным целым числом. Это пригодится позже, когда мы будем объявлять final нашего Activity, чтобы дочерний Activity мог вернуть результат.

Если мы сейчас запустим код и нажмем на кнопку, то получим следующее сообщение об ошибке:



Изображение 20. Activity не объявлен в AndroidManifest.xml.

Примечание: *некоторые сообщения об ошибках несут в себе весьма важную информацию, так что их стоит прочесть!*

Так что теперь мы должны объявить наш Activity “MySecondActivity” в AndroidManifest.xml. Для этого после </activity> (закрытия первого тега) мы пишем:

```
<activity android:name=".MySecondActivity"
            android:label="@string/app_name">
      <intent-filter>
            <action
                  android:name="android.intent.action.VIEW" />
            <category
                  android:name="android.intent.category.DEFAULT" />
      </intent-filter>
</activity>
```

На этот раз мы не выбирали ”MAIN” для action и “LANCHER” для category, так как нам нафиг не надо, чтобы этот Activity запускался при старте приложения.

“MySecondActivity” теперь доступен при щелчке по кнопке, но он так же обращается к main.xml:

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

Поэтому сейчас мы должны создать дополнительный layout-файл для “MySecondActivity”. Он должен содержать EdiText (на языке Swing – TextField) и кнопку для возвращения к главному Activity.

```
<?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" >
<EditText
      android:id="@+id/et_keyword"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"/>
<Button android:id="@+id/btn_return"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Submit" />
</LinearLayout>
```

Конечно, оба они, и Button, и EditText должны иметь android:id, чтобы мы могли обратиться к ним из кода. Сейчас мы изменим SetContentView “MySecondActivity” на:

```
/* IhemqZ EditText baXML-nZceZ*/
this.et_keyword = (EditText)this.findViewById(R.id.et_keyword);

/* IhemqZ_f dghidm ^ey \ha\jZs_qb-ynZaeZ*/
Button b = (Button)this.findViewById(R.id.btn_return);
/* Hlke_`b\Z_f debd ih dghid_
b.setOnClickListener(new OnClickListener(){
      public void onClick(View arg0) {
            /* Wlhl dh^ \uihegblky ijb s_eqd_ ih.dghid_
      }
});
```

Возвращение значения от дочернего Activity

Вернуть значение от дочерних Activity к вызвавшему их главному Activity так же очень просто:

```
/* IhemqZ_f \\_^_gghEditText k id=et_keyword agZq_gb/
String keyword = MySecondActivity.this.et_keyword.getText().toString();
/* lZjZf_lj i_j_^Zgguc wlhc nmgdpbb
 * [m^_l ^hklmiActivity, kha^Z\r_fm wActivity */
MySecondActivity.this.setResult(1, keyword);
/* GZrActivity \uiheg_gp fh`_l[ulv aZjul */
MySecondActivity.this.finish();
```

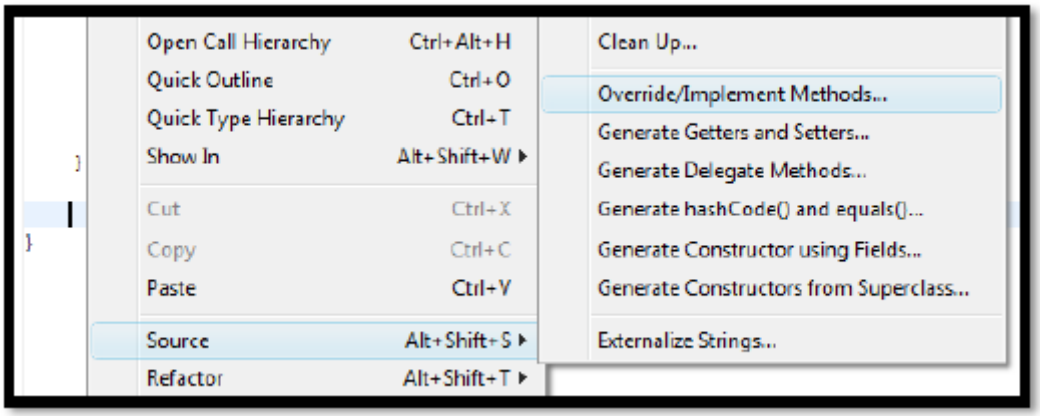
Можно было бы дополнительно отправить назад так называемый Bundle (что является более или менее HashMap), но я Вам об этом расскажу позже.

Понятно, что главный Activity должен отследить тот факт, что его дочерняя деятельность вернула результат. Чтобы достигнуть этого, мы должны переписать одну функцию из класса Activity, называющуюся onActivityResult(…):

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                String data, Bundle extras) {
      /* Dh^  ihf_s_gguc kx^Z  \uihegblky ijb \ha\jZr_gbb j_anvivlZ*/
      super.onActivityResult(requestCode, resultCode, data, extras);
}
```

Вы, возможно, увидели, что первый параметр тут – requestCode – тот самый requestCode, который мы передали Activity ранее. Так что, если у нас есть больше чем один дочерний Activity, понять, какой же вернулся, мы можем, проверив этот параметр.

Совет: *у Eclipse есть функция, позволяющая получить список функций, которые можно переписать, расширяя базовые классы! Это очень полезная функция!*



Изображение 21. Использование Eclipse для поиска методов, которые можно переписать.

Теперь мы можем просто проверить requestCode и создать новый Intent для поиска в гугле:

```
/* Ijh\_jbf  dZdhc ^hq_jgbc Activity \_jgmky. (A^_kv fu bf__f lhevdh h^gh]h
switch(requestCode){
      case MYSECONDACTIVITY_REQUESTCODE:
            /* Kha^Z^bgfh\uc Intent ^ey
             * ihdZaZ ]m]eh\kdhc kljZgbpu ihbkdZ
             * k \ha\jZs_gguf dexq_\uf ke*h\hf
            Intent webIntent = new Intent(
                  android.content.Intent.VIEW_ACTION,
                  Uri.parse("http://www.google.com/search?q=" + data));
            startActivity(webIntent); break; }
}
```

Вы, возможно, увидели, что на этот раз мы создаем Intent иначе. В прошлый раз мы писали: “Хотим запустить XYZ.class”, но на этот раз мы описываем, что хотели бы сделать. Мы хотим посетить URI (Uniform Resourse Indicator), который так же может быть собран с использованием Uri.fromParts(…) .

Наше приложение теперь может запускать дочернюю деятельность, брать результат, введенный пользователем, передавать его главной деятельности и открывать оттуда браузер. Но если мы захотим передать данные дочерней деятельности? Как нам это сделать?

Передача данных дочерним Activity

В передаче данных дочерним Activity нет ничего сложного. Один из вариантов – использовать так называемые Bundle (Пакеты), которые являются более или менее обычным HashMap, который может взять только обычные типы данных и нести информацию от одного Activity другому. Помните, где мы запустили нашу дочернюю деятельность в предыдущей главе?

```
/* Kha^ZgbIntent ^eyklZjlZ
 * MySecondActivity. */
Intent i = new Intent(
      StartingSubactivities.this,
      MySecondActivity.class);
/* HlijZ\ey_fIntent HK qlh[u
 * kdZaZlv  qlh fu ohlbf aZimklblv
 * MySecondActivity dZd^hq_jg_]Activity. */
startSubActivity(i, 0x1337);
```

Теперь между этими двумя строками мы поместим вот что:

```
/* Kha^Z_bfundle, dhlhjuc [m^_l
 * ijbkh[Zq_gIntent, ^ey^hklZ\db
 * bgnhjfZpbb ^hq_jgActivity. */
Bundle b = new Bundle();
b.putString(MY_DEFAULTSTRING_ID, "anddev.org");
/* ljbkh[Zqb\Z_bfundle d gZr_fmIntent. */
i.putExtras(b);
```

Где MY_DEFAULTSTRING_ID – любой индефикатор строки, с помощью которого потом ее можно извлечь из Bundle. Также можно использовать метод Intent.putExtra(...,...), который берет только одну информацию (one information) в каждый запрос. Его мы, возможно, будем использовать позже.

Сейчас нам надо извлечь информацию в нашем дочернем Activity. Каждый Activity может получить доступ к Intent, который запустил его, вызвав getIntent(). Если Bundle был присобачен к Activity, мы можем извлечь его, используя getIntent().getExtras(). В этом случае мы забьем полученную из Intent строку в поле ввода, куда пользователь вводит слова для поиска:

```
  «
setContentView(R.layout.second);
/* Ihemq Z_Bfndle hl Intent,
 * dhlhjuca Zimklbwlhl Activity.*/
Bundle bundle = getIntent().getExtras();
/* Ba\e_q_f kljhdm
 * bkihevamy,
 * dhlhjuc fu hij_^_ebeb j.Zg/__
String default_keyword = bundle.getString(
      StartingSubactivities.MY_DEFAULTSTRING_ID);

/* Ihemqbf EditText ba XML-nZce.Z*/
this.et_keyword = (EditText)this.findViewById(R.id.et_keyword);

/* AZ[v_f ihemq_ggmx kljhdm EditText. */
this.et_keyword.setText(default_keyword);

/* Ihemqbdghidm \ha\jZlZ XMLa-nZce.Z*/
  «
```

Так давайте вспомним, что Вы изучили или сделали до сих пор:
- Создали простое Приложение, используя XML Layout.
- Сделали то же самое через java Layout.
- Запустили Андроидское Приложения.
- Использовали LogCat для того, чтобы отладить работу приложения.
- Использовали Intent для запуска дочерних Activity.
- Нашли View, определенные в XML, чтобы использовать их в Java коде.
- Отловили клик по View.
- Вернули значение от дочернего Activity.
- Передали значение дочернего Activity.

Не плохо для начала.

Важные Layout и ViewGroup

Андроид обеспечивает много полезных Layout, кроме LinearLayout, который мы использовали до сих пор. Самые важные Layout, кроме LinearLayout, вот:

- RelativeLayout – View родственны друг другу (Views are placed relative to each other).
- TableLayout – Используй таблицы, как в HTML.

Дополнительно поделюсь с Вами наиболее часто используемыми мной ViewGroup:

- GridView – подобен TableLayout, поддерживает “Адаптеры”.
- ScrollView – если Ваше приложение велико, Вы можете листать его.
- TabHost – содержимое экрана табулировано (Displays content in tabs).

RelativeLayout

Используя RelativeLayouts, Вы определяете позицию View относительно других View-соседей.

Библиография

1. **Open Hanset Alliance.** Open Hanset Alliance. [Online]
http://www.openhandsetalliance.com/.

2. **GoogleGroups.** Android Discussion Groups. [Online]
http://code.google.com/android/groups.html.

3. **Alexa Webstats.** The Web Information Company. *alexa.com.*[Online]
http://www.alexa.com.

4. **Gramlich, Nicolas.** Android Development Community | Android Tutorials. [Online]
http://anddev.org.

5. **Hobbs, Zach.** Hello Android. [Online]
http://helloandroid.com/.

6. **Nguyen, Vincent.** Android Community. [Online]
http://androidcommunity.com/.

7. **Srinivas, Davanum.** Show me the code! [Online]
http://davanum.wordpress.com/.

(Тут в конце еще какая-то фигня, как я понял, ссылки на их буржуйские сайты. Там небось все по-английски, или еще хуже, так что интересно это только тем, кто его хорошо знает. Ну, а уж тот, надеюсь, и сам это переведет, а то я что-то устал, ну его… - прим. переводчика)
(Никакого послесловия тут нет, только ссылки – ПРОМТ гарантирует - прим. переводчика).

Lost chapters

The communities

From the first day on, a couple of communities arose from the depth of the internet. The most important community-website to mention (*depending on Alexa-stats* (3)) is http://anddev.org (4) , which is the most viewed Tutorial/Community-Board for Google Android up to date. Also well known are the Community-Boards: http://helloandroid.com (5) and http://androidcommunity.com (6). Last but definitely not least to mention is the http://davanum.wordpress.com -Blog (7), which provided some very early sample code where many programmers found a good first source.

Some sites were more, some less promising, so sooner or later the viewers pointed their focus to the more-important sites listed above.