

Содержание (настоящее)

Введение

Ваш мозг думает об Android. Вы сидите за книгой и пытаетесь что-нибудь выучить, но ваш мозг считает, что вся эта писанина не нужна. Ваш мозг говорит: «Выгляни в окно! На свете есть более важные вещи, например сноуборд». Как заставить мозг думать, что ваша жизнь действительно зависит от умения разрабатывать приложения для Android?

Для кого написана эта книга?	30
Мы знаем, о чем вы думаете	31
И мы знаем, о чем думает ваш мозг	31
Метапознание: наука о мышлении	33
Вот что сделали МЫ	34
Примите к сведению	36

Как бы теперь
заставить мой мозг
все это запомнить...



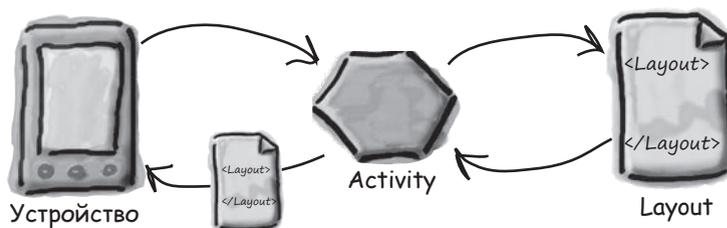
1 Первые Шаги

С головой в пучину

Система Android покорила мир. Все хотят иметь планшет или смартфон, а устройства на базе Android пользуются невероятной популярностью. В этой книге мы научим вас разрабатывать собственные приложения, а также покажем, как построить простое приложение и запустить его на виртуальном устройстве Android. Попутно будут рассмотрены основные компоненты приложений Android — такие, как активности и макеты. Все, что от вас потребуется, — некоторые базовые знания Java...



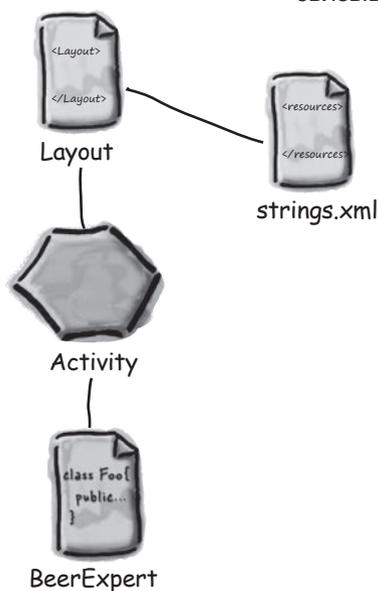
Добро пожаловать в мир Android	40
Платформа Android в разрезе	41
Вот что мы сейчас сделаем	42
Среда разработки	43
Установка Android Studio	44
Построение простого приложения	45
Как построить приложение	46
Активности и макеты: с высоты птичьего полета	50
Только что вы создали свое первое Android-приложение	53
Android Studio создает всю структуру папок за вас	54
Полезные файлы в проекте	55
Создание виртуального устройства Android	62
Запуск приложения в эмуляторе	65
Информация о ходе запуска отображается на консоли	66
Что же только что произошло?	68
Модификация приложения	69
Что содержит макет?	70
activity_main.xml состоит из двух элементов	71
Обновление текста, выводимого в макете	72
Ваш инструментарий Android	74



2 Построение интерактивных приложений

Приложения, которые что-то делают

Обычно приложение должно реагировать на действия пользователя. В этой главе вы узнаете, как существенно повысить интерактивность ваших приложений. Мы покажем, как заставить приложение делать что-то в ответ на действия пользователя и как заставить активность и макет общаться друг с другом, как старые знакомые. Попутно вы больше узнаете о том, как на самом деле работает Android; мы расскажем о R — неприметном сокровище, которое связывает все воедино.

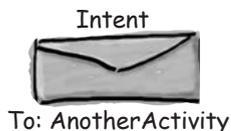


Строим приложение для выбора пива	76
Создание проекта	78
Мы создали активность и макет по умолчанию	79
Знакомство с визуальным редактором	80
Добавление кнопки в визуальном редакторе	81
В activity_find_beer.xml появилась новая кнопка	82
Подробнее о коде макета	83
Посмотрим, что же получилось	87
Жестко запрограммированный текст усложняет локализацию	88
Создание строковых ресурсов	89
Использование строкового ресурса в макете	90
Код activity_find_beer.xml	91
Добавление значений в список	94
Добавление string-array в strings.xml	95
Тест-драйв раскрывающегося списка	96
Кнопка должна что-то делать	97
Как заставить кнопку вызвать метод	98
Как выглядит код активности	99
Добавление в активность метода onClickFindBeer()	100
Метод onClickFindBeer() должен что-то делать	101
Получив ссылку на объект View, вы можете вызывать его методы	102
Обновление кода активности	103
Первая версия активности	105
Построение вспомогательного класса Java	108
Что происходит при выполнении кода	112
Ваш инструментарий Android	114

Множественные активности и интенты

Предъявите свой интент

Для большинства приложений одной активности недостаточно. До настоящего момента мы рассматривали приложения с одной активностью; для простых приложений это нормально. Однако в более сложной ситуации одна активность попросту не справляется со всеми делами. Мы покажем вам, как строить приложения с несколькими активностями и как организовать взаимодействие между активностями с использованием интентов. Также вы узнаете, как использовать интенты за пределами приложения и как выполнять действия при помощи активностей других приложений на вашем устройстве. Внезапно перед вами открываются совершенно новые перспективы...

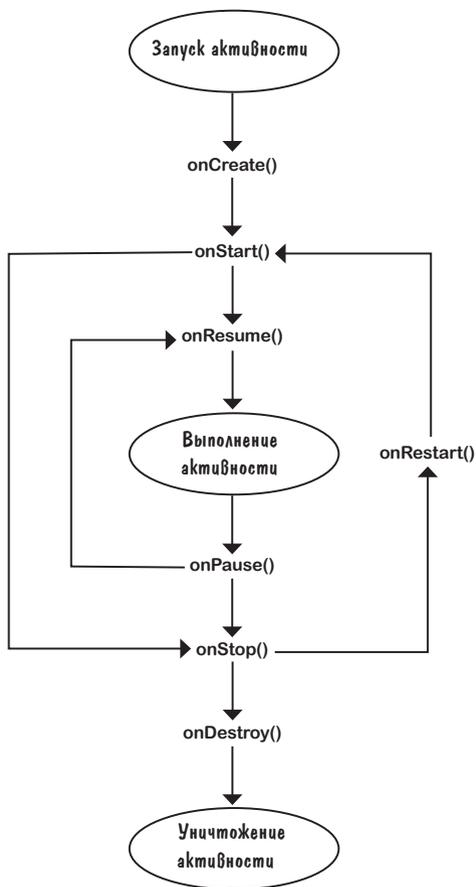


Приложение может содержать несколько активностей	116
Структура приложения	117
Создание проекта	117
Обновление макета	118
Создание второй активности и макета	120
Знакомьтесь: файл манифеста Android	122
Интент – разновидность сообщения	124
Что происходит при запуске приложения	126
Передача текста второй активности	128
Обновление кода CreateMessageActivity	133
Приложение можно изменить так, чтобы сообщения отправлялись другим людям	136
Как работают приложения Android	137
Создание интента с указанием действия	139
Изменение интента для использования действия	140
Что происходит при выполнении кода	141
Фильтр интентов сообщает Android, какие активности могут обработать те или иные действия	143
Как Android использует фильтр интентов	144
Запуск приложения на РЕАЛЬНОМ устройстве	147
А если вы хотите, чтобы пользователь ВСЕГДА выбирал активность?	150
Что произойдет при вызове createChooser()	151
Изменение кода создания активности	153
Если подходящих активностей НЕТ	155
Ваш инструментарий Android	156

4 Жизненный Цикл активности

Из жизни активностей

Активности образуют основу любого Android-приложения. Ранее вы видели, как создавать активности и как организовать запуск одной активности из другой с использованием интента. Но что при этом происходит, если заглянуть поглубже? В этой главе более подробно рассматривается жизненный цикл активностей. Что происходит при создании или уничтожении активностей? Какие методы вызываются, когда активность становится видимой и появляется на переднем плане, и какие методы вызываются, когда активность теряет фокус и скрывается? И как выполняются операции сохранения и восстановления состояния активности? В этой главе вы получите ответы на все эти вопросы.



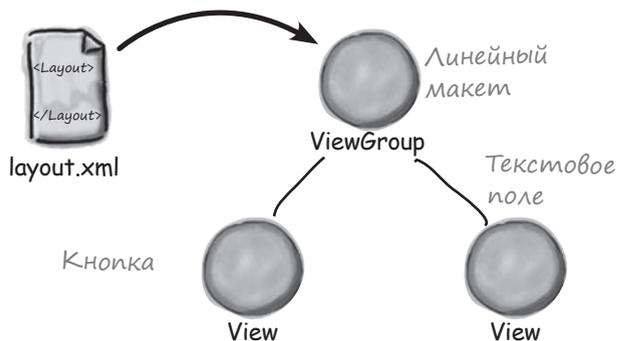
Как на самом деле работают активности?	158
Приложение Stopwatch	160
Добавление строковых ресурсов	161
Как работает код активности	163
Добавление кода кнопок	164
Метод runTimer()	165
Полный код runTimer()	167
Полный код StopwatchActivity	168
Поворот экрана изменяет конфигурацию устройства	174
Состояния активности	175
Жизненный цикл активности: от создания до уничтожения	176
Обновленный код StopwatchActivity	180
Что происходит при запуске приложения	181
Жизнь активности не ограничивается созданием и уничтожением	184
Обновленный код StopwatchActivity	189
Что происходит при запуске приложения	190
А если приложение видимо только частично?	192
Жизненный цикл активности: видимость	193
Прекращение отсчета времени при приостановке активности	196
Реализация методов onPause() и onResume()	197
Полный код StopwatchActivity	198
Что происходит при запуске приложения	201
Краткое руководство по методам жизненного цикла	205
Ваш инструментарий Android	206

5 Представления и Группы

Представление начинается

Вы уже видели, как происходит размещение компонентов графического интерфейса на экране в линейных макетах. Тем не менее это была лишь вершина айсберга. В этой главе мы заглянем поглубже, и вы узнаете, как на самом деле работают линейные макеты. Вы познакомитесь с компонентом **FrameLayout** — простым компонентом, предназначенным для размещения представлений. Также в этой главе будет представлен обзор **основных компонентов графического интерфейса и способов их использования**. К концу главы вы увидите, что несмотря на внешние различия, у всех макетов и компонентов графического интерфейса **больше общего, чем кажется на первый взгляд**

В композиционных макетах представления могут накладываться поверх друг друга. Например, это позволит вам вывести текст поверх графического изображения.

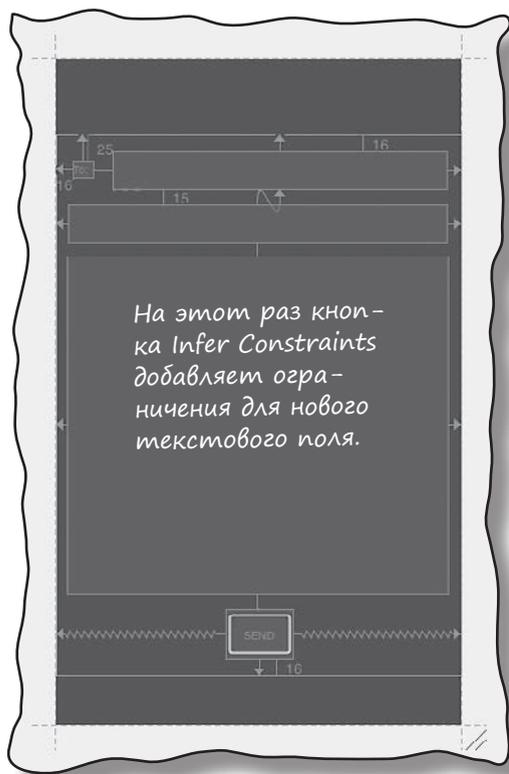


Пользовательский интерфейс состоит из макетов и компонентов графического интерфейса	208
LinearLayout отображает представления в строку или в столбец	209
Добавление файла ресурсов размеров для последовательного применения отступов между макетами	212
Создание интервалов между представлениями	214
Изменение базового линейного макета	215
Добавление весов	217
Атрибут gravity и положение содержимого в представлении	220
Полная разметка линейного макета	224
Вложенные макеты	229
Полная разметка вложения представлений	230
Знакомство с представлениями	239
Надпись	239
Текстовое поле	240
Кнопка	241
Двухпозиционная кнопка	242
Выключатель	243
Флажки	244
Переключатели	246
Раскрывающийся список	248
Графическое представление	249
Вывод изображений на кнопках	251
Прокручиваемые представления	253
Ваш инструментарий Android	258

6 Макеты с ограничениями

Расставить по местам

Давайте честно признаем: создать хороший макет не так просто. Это нужно уметь. Если вы строите приложения, которыми будут пользоваться люди, нужно позаботиться о том, чтобы они выглядели именно так, как было задумано. До сих пор мы показывали, как пользоваться линейными и композиционными макетами... Но что, если ваш макет имеет более сложную структуру? Для таких случаев в Android появилась новая возможность — макеты с ограничениями, разновидность макетов, которая обычно строится в визуальном режиме по схеме. Вы узнаете, как при помощи ограничений задавать позицию и размеры представлений независимо от размера экрана и ориентации. В завершение главы мы покажем, как сэкономить время, поручив Android Studio вычислить и добавить ограничения за вас.



Вложенные макеты бывают неэффективными	260
Макеты с ограничениями	261
Убедитесь в том, что в проект включена библиотека Constraint Layout Library	262
Добавление строчных ресурсов в файл strings.xml	263
Использование схемы	264
Позиционирование представлений с использованием ограничений	265
Добавление вертикального ограничения	266
Изменения на схеме отражаются в XML	267
Как выровнять представление по центру	268
Настройка позиции представления	269
Как изменить размеры представления	270
Выравнивание представлений	276
Построение реального макета	277
Сначала добавляется верхняя строка представлений	278
Среда разработки предполагает, какие ограничения нужно добавить в макет	279
На схему добавляется новая строка...	280
Остается добавить представление для сообщения	281
Ваш инструментарий Android	283

7

Списковые представления и адаптеры

Обо всем по порядку

Хотите знать, как лучше организовать Android-приложение? Мы рассмотрели основные структурные элементы, используемые при построении приложений; теперь пора привести знания в порядок. В этой главе мы покажем, как взять разрозненные идеи и превратить их в классное приложение. Мы покажем, как списки данных могут стать основой структуры вашего приложения и как связывание списков позволяет создавать мощные и удобные приложения. Попутно вы в общих чертах узнаете, как при помощи слушателей событий и адаптеров сделать ваше приложение более динамичным.

Вывести начальный экран со списком всех команд.

Вывести меню со всеми блюдами, которые может заказать посетитель..

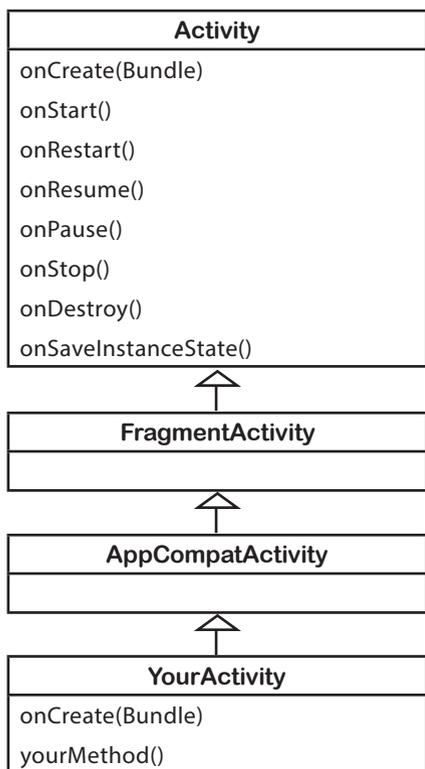
Вывести подробную информацию по каждому напитку.

Каждое приложение начинается с идей	286
Навигация с использованием списковых представлений	289
Построим приложение Starbuzz	290
Активность детализации с информацией о напитке	291
Структура приложения Starbuzz	292
Класс Drink	294
Использование спискового представления для вывода списка	297
Полная разметка макета верхнего уровня	298
Активность категории выводит данные, относящиеся к одной категории	305
Обновление файла activity_drink_category.xml	306
Для нестатических данных используйте адаптер	307
Связывание списковых представлений с адаптерами при помощи адаптера массива	308
Добавление адаптера массива в DrinkCategoryActivity	309
Как мы обрабатывали щелчки в TopLevelActivity	314
Полный код DrinkCategoryActivity	316
Обновление представлений	319
Код DrinkActivity	321
Что происходит при запуске приложения	322
Ваш инструментарий Android	326

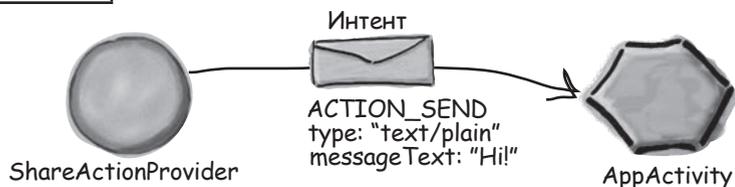
8 Библиотеки поддержки и панели приложений

В поисках короткого пути

Все мы предпочитаем короткие пути к цели. В этой главе вы узнаете, как ускорить выполнение команд ваших приложений при помощи панелей приложений. Вы узнаете, как запускать другие активности из элементов действий на панели приложения, как передавать данные другим приложениям при помощи провайдера передачи информации и как перемещаться в иерархии приложения с использованием кнопки Вверх на панели приложения. Заодно вы познакомитесь с библиотеками поддержки Android, с которыми ваши приложения будут выглядеть современно даже в старых версиях Android.



Хорошее приложение имеет четкую структуру	328
Типы навигации	329
Создание проекта Pizza	333
Добавление панели инструментов в макет...	348
...или определение панели инструментов в отдельном макете	349
Включение панели инструментов в макет активности	350
Добавление действий на панель приложения	353
Обновление activity_order.xml	354
Обновление OrderActivity.java	355
Изменение текста на панели приложения	356
Разметка AndroidManifest.xml	357
Управление внешним видом действия	360
Полный код MainActivity.java	363
Добавление кнопки Вверх	367
Передача информации с панели приложения	369
Добавление провайдера в файл menu_main.xml	370
Полный код MainActivity.java	372
Ваш инструментарий Android	375



9 Фрагменты

Модульная структура

Вы уже умеете создавать приложения, которые работают одинаково независимо от устройства, на котором они запускаются. Но что, если ваше приложение должно выглядеть и вести себя по-разному в зависимости от того, где оно запущено — на телефоне или планшете? В таком случае вам понадобятся фрагменты — модульные программные компоненты, которые могут повторно использоваться разными активностями. Мы покажем, как создавать простые фрагменты и списковые фрагменты, как добавлять их в активности и как организовать взаимодействие между фрагментами и активностями.



activity_detail

Ваше приложение должно хорошо смотреться на всех устройствах	378
Фрагменты дают возможность повторно использовать код	380
Версия приложения для телефона	381
Создание проекта и активностей	383
Добавление кнопки в макет MainActivity	384
Как добавить фрагмент в проект	386
Метод onCreateView() фрагмента	388
Включение фрагмента в макет активности	390
Взаимодействие фрагмента и активности	397
Класс Workout	398
Передача идентификатора фрагменту	399
Жизненный цикл фрагмента	403
Заполнение представлений в методе onStart() фрагмента	405
Создание фрагмента со списком	410
Обновленный код WorkoutListFragment	415
Разметка activity_main.xml	419
Связывание списка с детализацией	422
Код WorkoutListFragment.java	425
Передача идентификатора WorkoutDetailFragment	427
Ваш инструментарий Android	430

Списковый фрагмент содержит собственное списковое представление, так что вам не придется создавать его самостоятельно: достаточно предоставить списковому фрагменту данные.



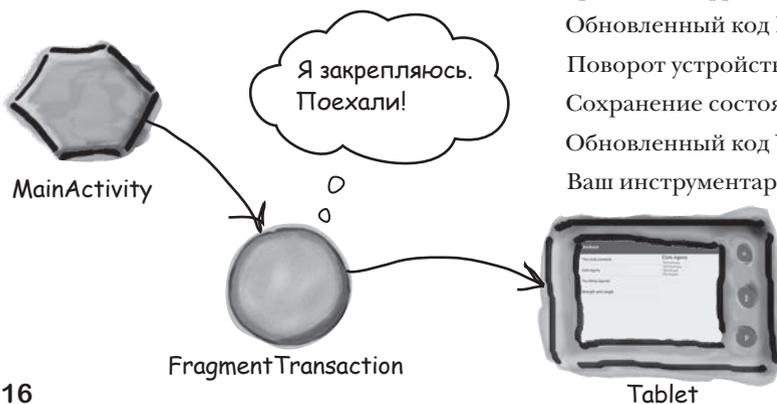
10 Фрагменты для больших интерфейсов

Разные размеры, разные интерфейсы

До сих пор наши приложения запускались только на устройствах с малыми экранами. Но что, если пользователи используют планшетные устройства? В этой главе вы увидите, как создать гибкий пользовательский интерфейс, чтобы ваше приложение выглядело и работало по-разному в зависимости от типа устройства, на котором оно запущено. В этой главе вы научитесь управлять поведением приложения с использованием кнопки Назад при помощи стека возврата и транзакций фрагментов. Наконец, вы узнаете, как сохранять и восстанавливать состояние фрагментов.



Приложение Workout одинаково выглядит на телефонах и планшетах	432
Проектирование интерфейса для больших экранов	433
Версия для телефона	434
Версия для планшета	435
Создание AVD для планшета	437
Размещение ресурсов для конкретного типа экрана в специальных папках	440
Выбор имен папок	441
Планшеты используют макеты из папки layout-large	446
Как работает код	448
Фрагменты должны работать с кнопкой Назад	451
Стек возврата	452
Проверка макета, используемого устройством	455
Обновленный код MainActivity	456
Транзакции фрагментов	457
Обновленный код MainActivity	461
Поворот устройства нарушает работу приложения	465
Сохранение состояния фрагмента...	467
Обновленный код WorkoutDetailFragment.java	468
Ваш инструментарий Android	470



11 Динамические фрагменты

Вложение фрагментов

До сих пор мы занимались созданием и использованием статических фрагментов. Но что, если вы хотите придать своим фрагментам немного динамики? У динамических фрагментов много общего с динамическими активностями, но есть и важные различия, которые необходимо учитывать. В этой главе вы научитесь преобразовывать динамические активности в рабочие динамические фрагменты. Вы узнаете, как использовать транзакции фрагментов для хранения состояния фрагмента. Наконец, мы покажем, как вложить один фрагмент в другой и как диспетчер дочерних фрагментов помогает решать проблемы с некорректным поведением стека возврата.

Когда я вижу атрибут `android:onClick`, я считаю, что он относится ко мне. Выполняются мои методы, а не методы фрагмента.



Активность

Создание динамических фрагментов	472
Новая версия приложения	474
Создание TempActivity	475
Класс TempActivity должен расширять AppCompatActivity	476
Код StopwatchFragment.java	482
Макет StopwatchFragment	485
Добавление фрагмента в макет TempActivity	487
Связывание OnClickListener с кнопками	495
Код StopwatchFragment	496
При повороте устройства показания секундомера обнуляются	500
Элемент <fragment> для статических фрагментов...	501
Перевод activity_temp.xml на использование FrameLayout	502
Полный код TempActivity.java	505
Включение фрагмента с секундомером в WorkoutDetailFragment	507
Полный код WorkoutDetailFragment.java	514
Ваш инструментарий Android	518

Я отображаю подробное описание комплекса упражнений, а также секундомер.

Транзакция для добавления StopwatchFragment вкладывается в транзакцию для добавления WorkoutDetailFragment.

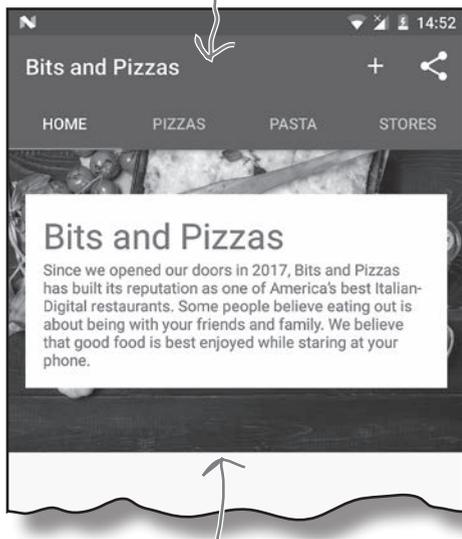


12 Design support library

Виджеты и жесты

Хотите разрабатывать приложения с полнофункциональным, современным интерфейсом? С выходом библиотеки Android Design Support Library разработчикам стало намного проще создавать приложения с современным пользовательским интерфейсом. В этой главе мы представим некоторые из ее ключевых аспектов. Вы научитесь создавать вкладки, чтобы пользователям было удобнее работать с системой навигации ваших приложений. Вы узнаете, как определить анимацию панелей инструментов, чтобы их можно было сворачивать или разворачивать по желанию пользователя. Вы научитесь добавлять плавающие кнопки действий для стандартных пользовательских действий. Наконец, мы познакомим вас с уведомлениями Snackbar — короткими содержательными сообщениями, с которыми может взаимодействовать пользователь.

Панель инструментов должна прокручиваться при прокрутке контента в TopFragment.



Этот контент добавляется в TopFragment.

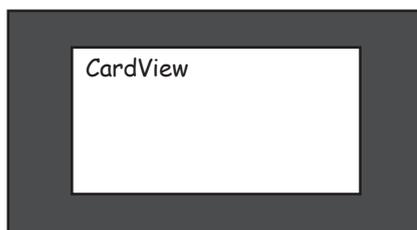
Возвращаемся к приложению Bits and Pizzas	520
Структура приложения	521
Использование компонента ViewPager для переключения между фрагментами	527
Включение ViewPager в макет MainActivity	528
Передача информации ViewPager о страницах	529
Код адаптера страничного компонента фрагментов	530
Полный код MainActivity.java	532
Добавление вкладок в MainActivity	536
Добавление вкладок в макет	537
Добавление вкладок в макет MainActivity	538
Связывание TabLayout с ViewPager	539
Полный код MainActivity.java	540
Реакция панели инструментов на прокрутку	546
Добавление CoordinatorLayout в макет MainActivity	547
Полная разметка fragment_top.xml	553
Добавление сворачивающейся панели инструментов в OrderActivity	555
Как создать простую сворачивающуюся панель	556
Размещение графики на панели инструментов	561
Обновленная разметка activity_order.xml	562
ФAB-кнопки и уведомления Snackbar	564
Обновленная разметка activity_order.xml	566
Полный код OrderActivity.java	571
Ваш инструментарий Android	573

13 RecyclerView и карточки

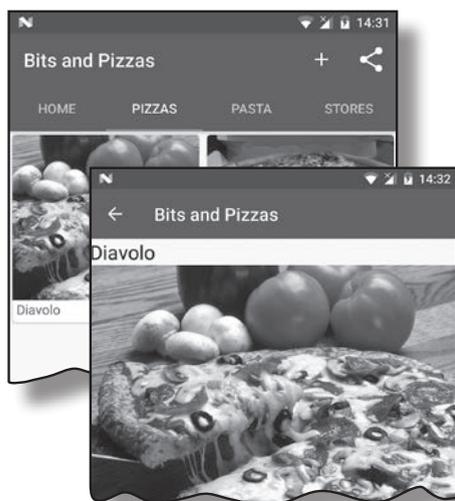
Переработка отходов

Вы уже видели, что скромное списковое представление играет ключевую роль во многих приложениях. Но по сравнению с компонентами материального оформления, которые были представлены выше, он слишком примитивен. В этой главе будет представлен компонент RecyclerView — более мощный списковый компонент, который обладает большей гибкостью и совмещается с принципами материального оформления. Вы научитесь создавать адаптеры, приспособленные к вашим данным, и сможете полностью изменить внешний вид списка всего двумя строками кода. Мы также покажем, как при помощи карточек имитировать в приложениях эффект трехмерного материального оформления.

ViewHolder



Каждый объект ViewHolder содержит CardView. Макет CardView был создан ранее в этой главе.



Работа над приложением Bits and Pizzas еще не закончена	576
Знакомство с RecyclerView	577
Добавление информации о пицце	579
Вывод данных пиццы в карточке	580
Создание представлений card view	581
Полная разметка card_captioned_image.xml	582
Как работает RecyclerView.Adapter	583
Создание адаптера RecyclerView	584
Определение класса ViewHolder	586
Полный код CaptionedImagesAdapter.java	589
Полный код CaptionedImagesAdapter.java (продолжение)	590
Создание RecyclerView	591
Включение RecyclerView в макет PizzaFragmen	592
Полный код PizzaFragment.java	593
Полный код PizzaFragment.java	596
Создание PizzaDetailActivity	605
Код PizzaDetailActivity.java	607
Реакция RecyclerView на щелчки	608
Прослушивание событий представлений в адаптере	609
Добавление интерфейса в адаптер	611
Код CaptionedImagesAdapter.java code (продолжение)	612
Реализация слушателя в PizzaFragment.java	613
Код PizzaFragment.java (продолжение)	614
Ваш инструментарий Android	616

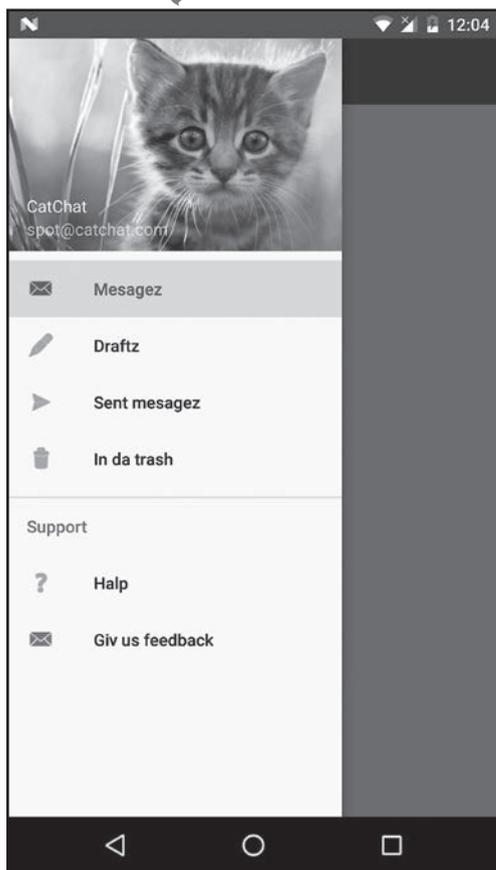
14

Выдвижные панели

Подальше положишь...

Как вы уже могли убедиться, вкладки сильно упрощают навигацию в приложениях. Но если вкладок *очень много* или вы захотите *разбить их на группы* — используйте навигационные выдвижные панели. В этой главе вы научитесь создавать навигационные панели, которые вызываются *из-за края активности одним прикосновением*. Вы узнаете, как назначить навигационной панели заголовок и как создать структурированное меню для перехода ко всем основным точкам приложения. Наконец, мы покажем, как создать слушателя, чтобы навигационная панель *реагировала на жесты*.

← Приложение CatChat.



Вкладки упрощают навигацию...	618
Мы создадим навигационную панель для нового почтового приложения	619
Подробнее о навигационных панелях	620
Создание проекта CatChat	622
Создание InboxFragment	623
Создание DraftsFragment	624
Создание SentItemsFragment	625
Создание TrashFragment	626
Создание макета панели инструментов	627
Создание HelpActivity	629
Создание FeedbackActivity	630
Построение навигационной панели	631
Создание заголовка навигационной панели	632
Полный код nav_header.xml	633
Группировка команд	636
Использование группы для первого раздела	637
Создание подменю для раздела	638
Полная разметка menu_nav.xml	639
Создание навигационной панели	640
Полная разметка activity_main.xml	641
Добавление InboxFragment в MainActivity	642
Добавление кнопки вызова панели	645
Реакция на выбор команд на навигационной панели	646
Реализация метода onNavigationItemSelected()	647
Полный код MainActivity.java	653
Ваш инструментарий Android	657

15 Базы данных SQLite

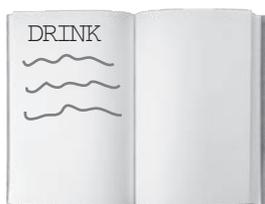
Работа с базами данных

Какая бы информация ни использовалась в приложении — рекордные счета или тексты сообщений в социальных сетях — эту информацию необходимо где-то хранить. В Android для долгосрочного хранения данных обычно используется база данных SQLite. В этой главе вы узнаете, как создать базу данных, добавить в нее таблицы и заполнить данными — все это делается при помощи удобных вспомогательных объектов SQLite. Затем будет показано, как выполнить безопасное обновление структуры базы данных и как вернуться к предыдущей версии в случае необходимости.



onCreate()

Помощник SQLite



База данных SQLite

Имя: "starbuzz"
Версия: 1

Возвращение в Starbuzz	660
Android хранит информацию в базах данных SQLite	661
Android включает классы SQLite	662
Текущая структура приложения Starbuzz	663
Переход на работу с базой данных	664
Помощник SQLite управляет базой данных	665
Создание помощника SQLite	666
Внутри базы данных SQLite	668
Таблицы создаются командами SQL	669
Вставка данных методом insert()	670
Вставка нескольких записей	671
Код StarbuzzDatabaseHelper	672
Что делает код помощника SQLite	673
А если структура базы данных изменится?	674
Номера версий баз данных SQLite	675
Что происходит при изменении номера версии	676
Обновление записей методом onUpgrade()	678
Метод onDowngrade()	679
Модификация базы данных	680
Обновление существующей базы данных	683
Обновление записей методом update()	684
Определение условий по нескольким столбцам	685
Изменение структуры базы данных	687
Удаление таблиц	688
Полный код помощника SQLite	689
Ваш инструментарий Android	694

16

Курсоры

Получение данных

Как же подключиться из приложения к базе данных SQLite?

В предыдущей главе было показано, как создать базу данных SQLite с использованием помощника SQLite. Пора сделать следующий шаг — узнать, как работать с базой данных из активностей. Эта глава посвящена чтению данных из базы. Вы узнаете, как использовать курсоры для получения информации из базы данных, как перемещаться по набору данных с использованием курсора и как получить данные из курсора. Затем мы покажем, как использовать адаптеры курсоров для их связывания со списковыми представлениями.

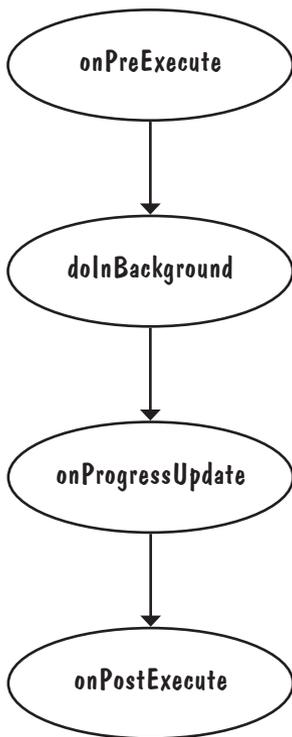
Чего мы добились...	696
Новая структура приложения Starbuzz	697
Изменения DrinkActivity для использования базы данных Starbuzz	698
Текущий код DrinkActivity	699
Получение ссылки на базу данных	700
Курсоры и чтение информации из базы данных	701
Выборка всех записей из таблицы	702
Упорядочение данных в запросах	703
Выборка по условию	704
И снова код DrinkActivity	707
Переходы между записями	709
Чтение данных из курсора	710
Последний шаг: закрытие курсора и базы данных	710
Код DrinkActivity	711
Что нужно сделать с DrinkCategoryActivity для использования базы данных Starbuzz	714
Текущий код DrinkCategoryActivity	715
Получить ссылку на базу данных Starbuzz...	716
...затем создать курсор, возвращающий данные	716
Как заменить данные массива в ListView?	717
Простой адаптер курсора связывает данные курсора с представлениями	718
Закрытие курсора и базы данных	720
История продолжается	721
Обновленный код DrinkCategoryActivity	726
Ваш инструментарий Android	729



17

Курсоры и асинхронные задачи Выполнение в фоновом режиме

В большинстве приложений данные должны обновляться. Вы научились создавать приложения, читающие данные из баз данных SQLite. Но что, если данные приложения должны обновляться? В этой главе вы узнаете, как научить приложение реагировать на ввод данных пользователем и обновлять значения в базе данных. Также вы узнаете, как обновлять содержимое экрана после модификации данных. В завершение мы покажем, как написание эффективного многопоточного кода с объектами AsyncTask ускоряет работу приложений.



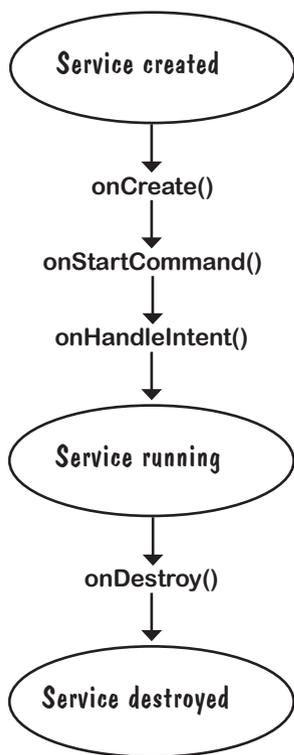
Обновление данных в приложении Starbuzz	732
Начнем с обновления DrinkActivity	733
Включение флажка в макет DrinkActivity	734
Вывод значения столбца FAVORITE	735
Полный код DrinkActivity.java	739
Вывод любимых напитков в TopLevelActivity	743
Переработка TopLevelActivity.java	745
Новый код TopLevelActivity.java	748
Изменение курсора методом changeCursor()	753
Обновленный код TopLevelActivity.java	754
Какой код для какого потока?	761
Класс AsyncTask выполняет асинхронные задачи	762
Метод onPreExecute()	763
Метод doInBackground()	764
Метод onProgressUpdate()	765
Метод onPostExecute()	766
Параметры класса AsyncTask	767
Полный код UpdateDrinkTask	768
Полный код DrinkActivity.java	770
Схема работы с объектами AsyncTask	775
Ваш инструментарий Android	775



18

Службы К вашим услугам

Существуют операции, которые должны выполняться постоянно, какое бы приложение ни обладало фокусом. Например, если вы запустили воспроизведение музыкального файла в приложении-проигрывателе, вероятно, музыка не должна останавливаться при переключении на другое приложение. В этой главе вы узнаете, как использовать службы — компоненты, выполняющие операции в фоновом режиме, научитесь создавать службы при помощи класса `IntentService`. Также мы разберемся в том, как жизненный цикл служб связан с жизненным циклом активности. Заодно вы научитесь регистрировать сообщения и держать пользователей в курсе дел с использованием встроенной службы уведомлений Android.

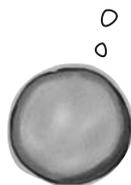


Службы работают незаметно для пользователя	778
ЗАПУСКАЕМАЯ служба	779
Использование класса <code>IntentService</code> для создания простой службы	780
Запись сообщений в журнал	781
Полный код <code>DelayedMessageService</code>	782
Объявление служб в <code>AndroidManifest.xml</code>	783
Добавление кнопки в <code>activity_main.xml</code>	784
Службы запускаются вызовом <code>startService()</code>	785
Что происходит при запуске приложения	786
Состояния запускаемой службы	788
Жизненный цикл запускаемых служб: от создания к уничтожению	789
Служба наследует методы жизненного цикла	790
В Android имеется встроенная служба уведомлений	793
Мы используем уведомления из библиотеки поддержки <code>AppCompat</code>	794
Создание построителя уведомлений	795
Добавление действия для определения активности, запускаемой по щелчку	796
Выдача уведомлений с использованием встроенной службы	797
Полный код <code>DelayedMessageService.java</code>	798
Что происходит при выполнении кода	800
Ваш инструментарий Android	803

19

Связанные службы и разрешения Связаны вместе

Запускаемые службы отлично подходят для фоновых операций — а если вам нужна служба с большей интерактивностью? В этой главе вы научитесь создавать связанные службы — разновидность служб, с которыми могут взаимодействовать ваши активности. Вы узнаете, как выполнить привязку к службе и как отменить ее после завершения работы для экономии ресурсов. Служба позиционирования Android поможет вам получать информацию местонахождения от GPS-приемника вашего устройства. Наконец, вы научитесь пользоваться моделью разрешений Android, включая обработку запросов разрешений во время выполнения.



OdometerService

Связанные службы привязываются к другим компонентам	806
Создание новой службы	808
Реализация IBinder	809
Добавление метода getDistance()	810
Обновление макета MainActivity	811
Создание объекта ServiceConnection	813
Применение bindService() для связывания	816
Вызов метода getDistance()	818
Полный код MainActivity.java	819
Что происходит при выполнении кода	821
Состояния связанных служб	825
Добавление библиотеки поддержки AppCompat	828
Добавление слушателя в OdometerService	830
Обновленный код OdometerService	833
Вычисление пройденного расстояния	834
Полный код OdometerService.java	836
Запрос разрешения	840
Выдача уведомление при отказе	844
Добавление кода уведомлений в onRequestPermissionsResult()	847
Полный код MainActivity.java	849
Ваш инструментарий Android	853
Надеемся, вы хорошо провели время в мире Android.	854