3.9. Панель инструментов и меню в Android приложениях

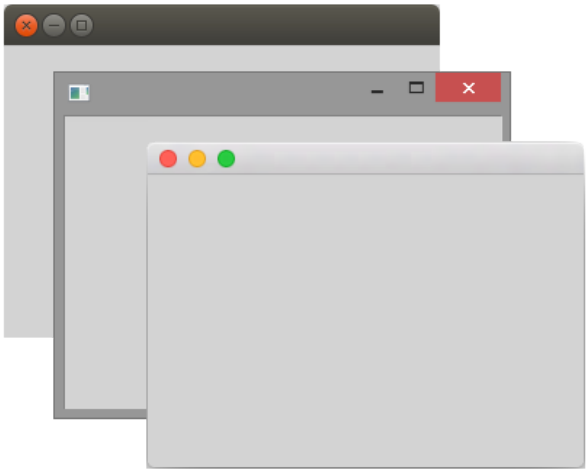
Сайт: Samsung Innovation Campus
Курс: Мобильная разработка на Kotlin
Книга: 3.9. Панель инструментов и меню в Android приложениях

Напечатано:: Murad Rezvan
Дата: понедельник, 3 июня 2024, 17:51

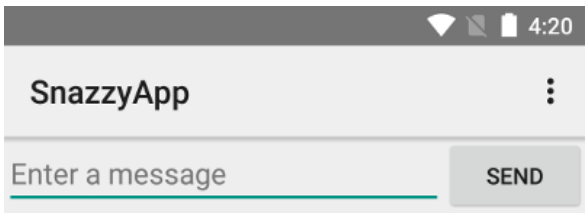
Оглавление

3.9.1. Визуальные NavigationUI элементы

В Android разработке сложилась определенная терминология относительно графического элемента - паннели в верху активности. Посмотреть описание этого элемента можно на сайте материал дизайна. Этот элемент по своему основному назначению соотвествует заголовку окна в настольных системах (Window Title) и в Android имеет разные названия AppBar, ActionBar, ToolBar, что иногда приводит к путанице. Ниже рассмотрим эволюционирование терминологии.

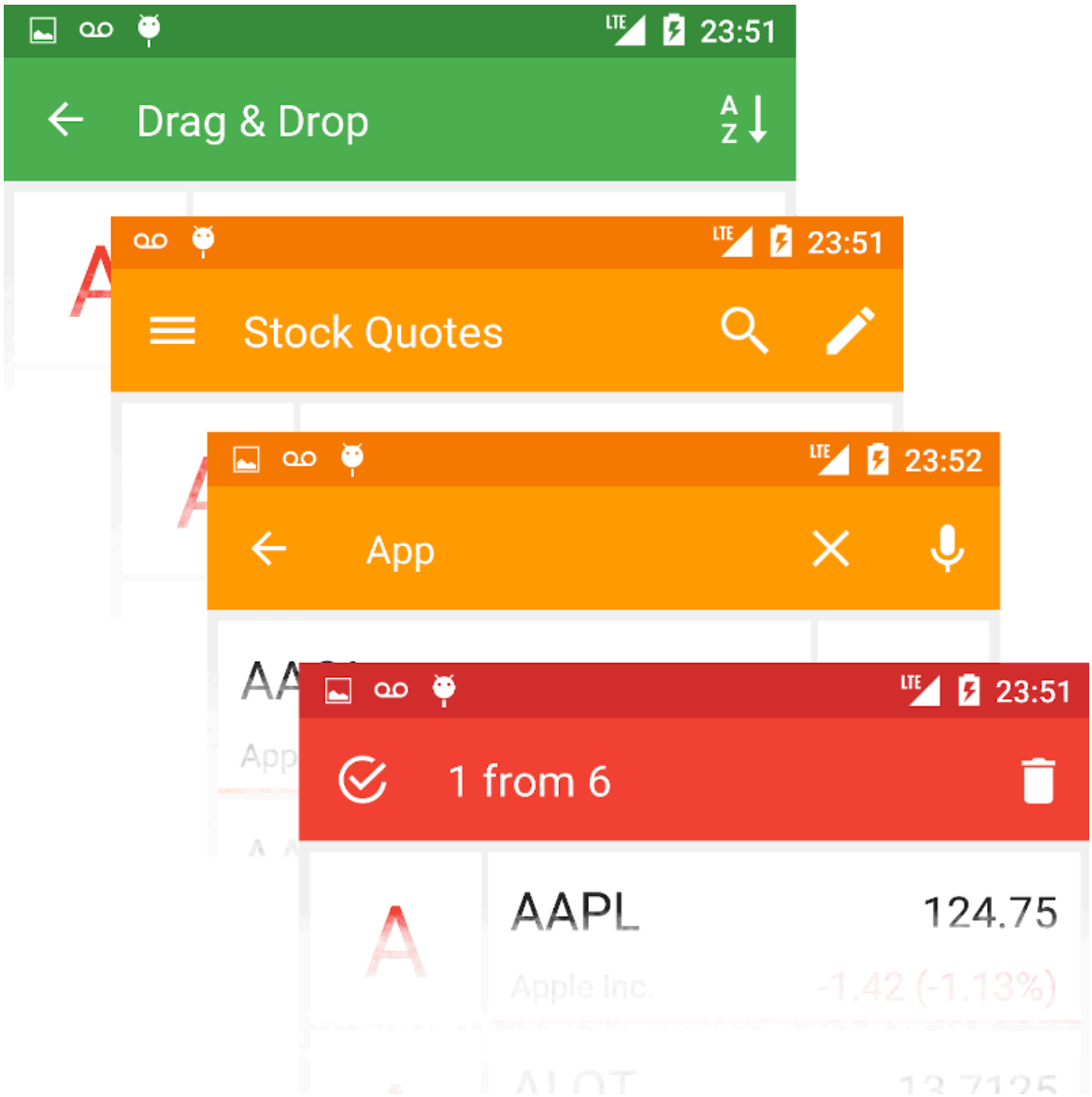


В самых первых Android появился AppBar и в самом простом виде на панели приложения отображается название активности с одной стороны и дополнительное меню с другой. Даже в этой простой форме панель приложений предоставляет пользователям полезную информацию и помогает придать приложениям Android единообразный вид.



Начиная с Android 3.0 (уровень API 11), все действия, использующие тему по умолчанию, имеют ActionBar в качестве панели приложения. Однако, по мере появления новых версий API, новые функции постепенно добавлялись во встроенную панель ActionBar. В результате нативный ActionBar ведет себя по-разному в зависимости от того, какую версию системы Android может использовать устройство. Напротив, самые последние функции добавляются в новую компоненту ToolBar библиотеки поддержки (SupportLibrary), и они доступны на любом устройстве, которое может использовать библиотеку поддержки.

По этой причине рекомендуется использовать класс ToolBar из библиотеки поддержки для реализации панелей приложений создаваемых активностей. Использование панели инструментов библиотеки поддержки помогает обеспечить единообразное поведение приложения на самом широком спектре устройств. Например, виджет панели инструментов предоставляет возможности материального дизайна на устройствах под управлением Android 2.1 (уровень API 7) или более поздней версии. В тоже время наттивная компонента ActionBar не поддерживает материальный дизайн, если устройство не работает под управлением Android 5.0 (уровень API 21) или более поздней версии.



В соответсивмм с вышеприведенными рекомендациями разберем именно компоненту **ToolBar** . Для демонстрации интерфейсных компонентов на этом занятии лучше создать активность *New->Activity->BasicActivity* либо при создании проекта выбрать шаблон Basic Activity. При этом будет создана активность с **CoordinatorLayout** в котором размещен **ToolBar** и **FloatingActionButton**. Ниже более подробно рассмотрим компоненты из шаблона BasicActivity.

Рассматриваемые далее примеры входят в практическую работу, исходный код которой можно скачать по ссылке

CoordinatorLayout

В палитре элементов современного API большинство виджетов уже оснащены свойствами MaterialDesign. Отдельная роль в материальном дизайне отведена поведению виджетов на экране пользователя в соответствии с моделью поведения в реальном мире. Разработчики изложили концепцию анимации движения виджетов и их взаимодействия на сайте MaterialDesign. В этой концепции учитывается многие эффекты, имитирующие поведение реальных объектов. Как то:

- при движении одного виджета - он может отодвигать другой (другие),
- при расширении одного виджета, другие уменьшаются,
- один виджет может смещаться под другой,
- движение виджета могут иметь различную скорость и ускорение,
- изменение освещенности объектов в зависимости от их положения и перемещения,
- обработка тени,
- и т.д.

За обеспечении поведения экранных виджетов в соответствии с моделью поведения реального мира отвечают контейнеры в которых эти виджеты сгруппированы. Один из таких контейнеров является **CoordinatorLayout**, которой сами разработчики называют "сверхмощный FrameLayout". CoordinatorLayout предназначен для двух основных вариантов использования:

- для включения виджетов верхнего уровня,
- и в качестве контейнера для обеспечения взаимодействия между дочерними виджетами.

Дополнительно можно указать поведение для дочерних представлений CoordinatorLayout, тем самым обеспечивая много разных взаимодействий в пределах одного родителя, и эти виджеты также могут взаимодействовать друг с другом. Виджетам можно указывать поведение по умолчанию при использовании их в качестве дочерних элементов CoordinatorLayout , при помощи аннотации CoordinatorLayout.DefaultBehavior. Поведения могут использоваться для реализации различных взаимодействий и дополнительных модификаций макета, начиная от выдвижных ящиков и панелей и заканчивая отклоняемыми элементами и кнопками, которые прилипают к другим элементам по мере их перемещения и анимации.

Если создавать активность как BasicActivity, то CoordinatorLayout в макете активности уже будет использован как корневой Laout, в котором уже включен NavHostFragment , AppBarLayout и ToolBar. Собственно в этом шаблоне уже включены три основные контейнера современного Material Design. В шаблоне по умолчанию BasicActivity например есть FloatingActionButton (FAB), в поведении которого присутствуют элементы материального поведения - при нажатии меняется его высота над макетом, а также снизу выдвигается SnakBar и перемещается FAB и его тень. Ниже приведен образец макета по умолчанию:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

И класс активности выглядит следующим образом:

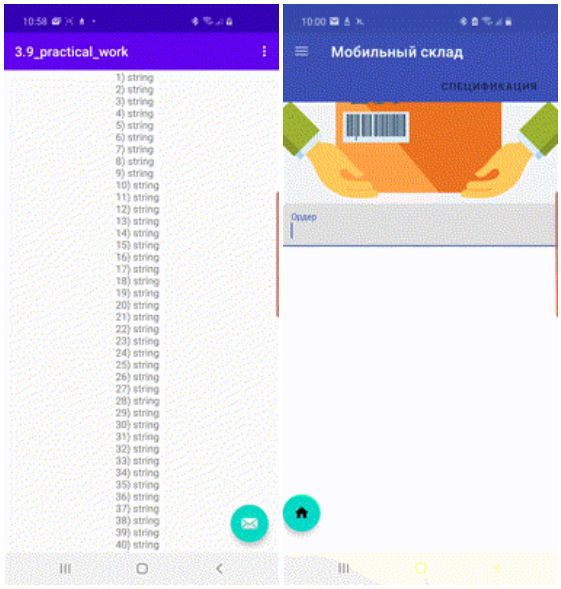
```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(findViewById(R.id.toolbar))

        findViewById<FloatingActionButton>(R.id.fab).setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show()
        }
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.menu_main, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_settings -> true
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

В коде, созданном по умолчанию, не так много возможностей, но основное четко продемонстрировано - CoordinatorLayout объединяет все элементы MaterialDesign, для обеспечения их правильной работы и отображения.

3.9.2. Дизайн верхней панели

AppBarLayout

Эта ViewGroup - это расширенный вариант LinearLayout, в котором реализовано MaterialDesign взаимодействий новых виджетов, позволяющих оформить макет в. современном стиле. Например, если немного изменить макет вышеуказанного приложения, то можно продемонстрировать эти улучшенные возможности, например прячущаяся верхняя панель при прокрутке содержимого:



Это полезная возможность, которая позволяет при начале просмотра содержимого окна показывать пользователю полную верхнюю панель, которая может содержать как название и элементы меню так и дополнительные элементы, например картинки, но при просмотре содержимого на экране ниже скрывать панель, максимально освобождая место для полезного контента. В коде это достигается буквально двумя аттрибутами макета.

Во первых в элемент ToolBar или AppBarLayout нужно добавить аттрибут

```
app:layout_scrollFlags="scroll|enterAlways"
```

А во вторых элемент макета ниже AppBarLayout должен быть скроллируемыми. Например RecyclerView или NestedScrollView

```
  <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">
...
</androidx.core.widget.NestedScrollView>
```

Указанная функциональность реализована в примере на github, в первом фрагменте.

Меню

OptionsMenu - это стандартное меню с дополнительными функциями. На старых андроид устройствах была отдельная кнопка Menu, нажатие которой вызывало это меню. В новых устройствах отдельную кнопку убрали, заменив на значок меню в виде трёх точек в правом верхнем углу. В шаблоне проекта BasicActivity уже присутствует готовый образец этого меню. Для этого автоматически создается файл с меню - res/menu/menu_main.xml .

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="ru.samsung.itschool.myapplication.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```

А в код активности добавляется функционал добавления меню на панель:

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.menu_main, menu)
    return true
}
```

Кроме того, необходимо добавить обработчик нажатий на пункты OptionalMenu

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
   Toast.makeText(applicationContext, "MenuItem click!", Toast.LENGTH_SHORT).show()
    return when (item.itemId) {
        R.id.action_settings -> true
        else -> super.onOptionsItemSelected(item)
    }
}
```

Таким образом выглядит вызов меню в рассматриваемом примере.

Контекстное меню

Помимо обычного OptionalMenu также в андроиде предусмотрена возможность подключения контекстного меню. Это примерный аналог контекстного меню в настольных системах, котоорое вызывается при нажатии на элемент интерфейса правой кнопки мыши. Контекстное меню вызываемое при длительном нажатии на объект (событие long-press). Также на некоторых устройствах контекстное меню может быть вызвано при нажатии трекбола или средней кнопки манипулятора D-pad. В отличие от обычного меню, в контекстном меню не поддерживаются значки и быстрые клавиши. Добавим в проект контекстное меню на TextView на втором фрагменте. Это меню можно "раздуть" из файла XML, но в этот раз, для разнообразия реализуем меню программно:

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?, menuInfo: ContextMenu.ContextMenuInfo?) {
    super.onCreateContextMenu(menu, v, menuInfo)
    //menuInflater.inflate(R.menu.menu_main, menu)
    menu!!.add(Menu.NONE, 101, Menu.NONE, "Открыть");
    menu!!.add(Menu.NONE, 102, Menu.NONE, "Сохранить");
}
```

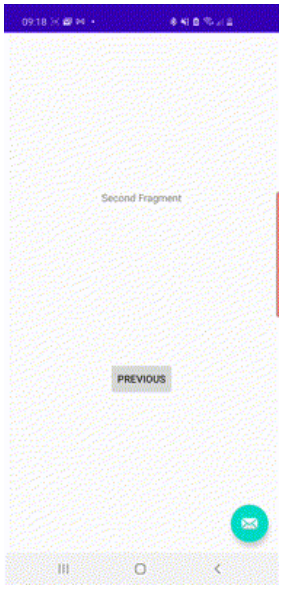
Аналогично обычному меню, необходимо реализовать функцию - обрабочик на нажатие в контекстном меню:

```
 override fun onContextItemSelected(item: MenuItem): Boolean {
    Toast.makeText(applicationContext, "ContextMenuItem click!", Toast.LENGTH_SHORT).show()
    return when (item.itemId) {
        R.id.action_settings -> true
        else -> super.onContextItemSelected(item)
    }
}
```

Однако в отличие от обычного меню нужно указать, на каком элементе будет подключено контестное меню. В рассматриваемом примере, на макете второго фрагмента присутсвует элемент **textview_second** и, сооветсвенно, в коде создания второго фрамента указываем, что на этот элемент разрешено открытие контестного меню:

```
val textview_second: TextView = view.findViewById(R.id.textview_second)
registerForContextMenu(textview_second)
```

Таким образом выглядит вызов контекстного меню на втором фрагменте рассматриваемого примера.



Контекстная панель инструментов

По аналоги с контекстным меню есть вариант работы верхней панели приложения в контекстном режиме. В этом режиме верхняя панель приложения трансформируется в контекстную панель действий для предоставления контекстных действий выбранным элементам. Например, после выбора пользователем фотографий из галереи верхняя панель приложения преобразуется в контекстную панель приложения с действиями, связанными с выбранными фотографиями.

Когда верхняя панель приложения превращается в контекстную панель действий, обычно происходят следующие изменения:

- Цвет бара меняется
- Значок навигации заменен на значок закрытия
- Текст заголовка верхней панели приложения преобразуется в текст контекстной панели действий
- Действия верхней панели приложения заменяются контекстными действиями
- После закрытия контекстная панель действий снова превращается в верхнюю панель приложения.

В рассматриваем примере создадим файл с элементами на контекстной панели и/или меню res/menu/context_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="ru.samsung.itschool.a3_9_practical_work.MainActivity">
    <item
        android:id="@+id/gallery"
        android:icon="@android:drawable/ic_menu_gallery"
        android:title="@string/gallery"
        android:contentDescription="@string/content_description_gallery"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="@string/search"
        android:contentDescription="@string/content_description_search"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/more"
        android:title="@string/more"
        android:contentDescription="@string/content_description_more"
        app:showAsAction="never" />
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```

Далее в обработчике нажатий FAB добавим перевод активности в режим контекстного AppBar - при помощи вызова метода startActionMode

```
        fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show()
            mActionMode= startActionMode(mActionModeCallback);
        }
```

Ну и конечно необходимо создать объект mActionModeCallback - слушатель для обработки событий контекстного AppBar-а

```
        mActionModeCallback = object: ActionMode.Callback {
            override fun onCreateActionMode(p0: ActionMode?, p1: Menu?): Boolean {
                p0!!.menuInflater.inflate(R.menu.context_menu,p1)
                return true
            }

            override fun onPrepareActionMode(p0: ActionMode?, p1: Menu?): Boolean {
                return false
            }

            override fun onActionItemClicked(p0: ActionMode?, p1: MenuItem?): Boolean {
                // здесь обрабатываются клики на элементы контекстного AppBar
                p0!!.finish()
                return false
            }

            override fun onDestroyActionMode(p0: ActionMode?) {

            }

        }
```

Начать тур для пользователя на этой странице