3.11. Вкладки

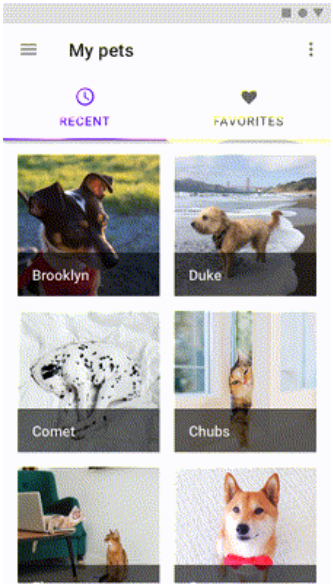
Сайт: Samsung Innovation Campus
Курс: Мобильная разработка на Kotlin
Книга: 3.11. Вкладки

Напечатано:: Murad Rezvan
Дата: понедельник, 3 июня 2024, 17:52

Оглавление

3.11.1 Навигация с помощью вкладок

Еще один элемент UI, позволяющий пользователю интуитивно понятно осуществлять навигацию - это механизм вкладок. Вкладки, появились в Android 5.0 (API 21). TabLayout обеспечивает горизонтальное расположение элементов для отображения вкладок, которые позволяют переключаться между различными фрагментами. Закладки - довольно удобный и эффектный элемент, котороый позволяет делать латеральные переходы по интерфейсу как при помощи кликов на закладках так и при помощищи свайпов, и даже при помощи например специальных манипулятор (например безель). Все необходимые классы добавлены в библиотеки поддержки и библиотеку material.



Для рассмотрения примера использования закладок используем готовый шаблон активности, который создается при помощи *New -> Activity -> TabbedActivity*.

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

В представленном шаблоне внутри CoordinatorLayout, который обеспечивает работоспособность компонентов материального дизайна, в самой нижней части AppBar добавляется элемент TabLayout

```
<com.google.android.material.tabs.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="?attr/colorPrimary" />
```

Фиксированное создание закладок в макете

В самом простом случае можно в элемент TabLayout вложить нужные закладки как элементы в макете дизайна следующим образом:

```
<com.google.android.material.tabs.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="?attr/colorPrimary">

            <com.google.android.material.tabs.TabItem
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/tab_text_1" />

            <com.google.android.material.tabs.TabItem
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/tab_text_2" />

        </com.google.android.material.tabs.TabLayout>
```

и конечно нужно в коде активности подключить обработчик выбора закладок, в котором вручную управлять установкой фрагментов на экран:

```
        tabs.addOnTabSelectedListener(object : TabLayout.OnTabSelectedListener {

            override fun onTabSelected(tab: TabLayout.Tab?) {
                // здесь переключаем фрагменты
                Toast.makeText(applicationContext,"Tab clicked:"+tab!!.text,Toast.LENGTH_SHORT).show()
            }

            override fun onTabReselected(tab: TabLayout.Tab?) {
            }

            override fun onTabUnselected(tab: TabLayout.Tab?) {
            }
        })
```

Представленный способ прост но достаточно негибок и в рассматриваемом примере не реализован. В используемом типовом шаблоне шаблоне Andorid Studio TabbedActivity для управления закладками используется класс **ViewPager**.

3.11.2 Использование класса ViewPager

Вместо указания фиксированного набора закладок в макете, TabLayout можно настроить с помощью ViewPager, чтобы:

- Динамически создавать элементы вкладок в зависимости от количества страниц, их заголовков и т. д.
- Синхронизировать выбранную вкладки и положения индикатора вкладки с эффектом пролистывания страницы.

Макет активности выглядит следующим образом:

```
<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/Theme.MyApplication.AppBarOverlay">

        <!-- Здесь дополнительный дизайн AppBar -->

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:tabMode="fixed"/>
    </com.google.android.material.appbar.AppBarLayout>

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

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

При использовании ViewPager нужно в классе активности выполнить три операции:

1. Создать объект-адаптер от класса наследника PagerAdapter, который связан с массивом строк названий закладок. В рассматриваемом примере в качестве адаптера используется объявленный класс SectionsPagerAdapter. Напомним, что обычно объект адаптер используется для возможности связи визуального объекта множественного типа (как RecyclerView, ViewPager, и т.д.) с перечисляемой структурой данных - в простейшем случае массивом . Не стал исключением и рассматриваемый пример.

2. Получить от Android объект ViewPager, view которого в данный момент присутствует на экране и установить в него адаптер, полученный на шаге 1.

3. Получить от Android объект TabLayout, view которого в данный момент присутствует на экране и установить в него viewPager, полученный на шаге 2.

```
    override fun onCreate(savedInstanceState: Bundle?) {
        // ...
        val sectionsPagerAdapter = SectionsPagerAdapter(this, supportFragmentManager)
        val viewPager: ViewPager = findViewById(R.id.view_pager)
        viewPager.adapter = sectionsPagerAdapter
        val tabs: TabLayout = findViewById(R.id.tabs)
        tabs.setupWithViewPager(viewPager)
        // ...
    }
```

Как говорилось выше PageAdapter - это классически класс-адаптер с типовым набором функциональности:

- получить количество объекта массива,
- по указанному индексу получить соответствующий ему фрагмент,
- по указанному индексу получить название фрагмента. Код выглядит следующим образом:

```
// массив с названиями фрагментов
private val TAB_TITLES = arrayOf(
        R.string.tab_text_1,
        R.string.tab_text_2
)

class SectionsPagerAdapter(private val context: Context, fm: FragmentManager)
    : FragmentPagerAdapter(fm) {

    override fun getItem(position: Int): Fragment {
        // получить фрагмент по указанному номеру
        return PlaceholderFragment.newInstance(position + 1)
    }

    override fun getPageTitle(position: Int): CharSequence? {
        // получить название фрагмента по указанному номеру
        return context.resources.getString(TAB_TITLES[position])
    }

    override fun getCount(): Int {
        // получить количество фрагментов/закладок
        return TAB_TITLES.size
    }
}
```

Класс PlaceholderFragment предназначен для создания нового фрагмента и ничем не примечателен. По сути это просто класс фрагмента, но в нем дополнительно есть статический метод newInstance для создания нового объекта этого класса. Его текст можно посмотреть в исходном коде практической работы. При выполнения рассматриваемого примера будет построена полностью рабочая система генерации фрагментов и подключения их в ViewPager при перелистывании закладок.

3.11.3 Новый класс ViewPager2

20 ноября 2019 года Google выпустила новую версию ViewPAger - Android ViewPager2. Более подробную информацию об этом релизе можно найти здесь. Переход от ViewPager к ViewPager2 описан здесь

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

Использование класса ViewPager2 внешне похоже на использование класса ViewPager, однако изменились сами имена используемых классов. Например в макете активности теперь:

```
 <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />
```

Класс адаптера теперь наследуется от FragmentStateAdapter, а не от FragmentPagerAdapter и в нем также изменилось именование методов:

```
class SectionsPagerAdapter(fa: FragmentActivity)
    : FragmentStateAdapter(fa) {

    override fun getItemCount(): Int = TAB_TITLES.size

    override fun createFragment(position: Int): Fragment =
        PlaceholderFragment.newInstance(position + 1)
}
```

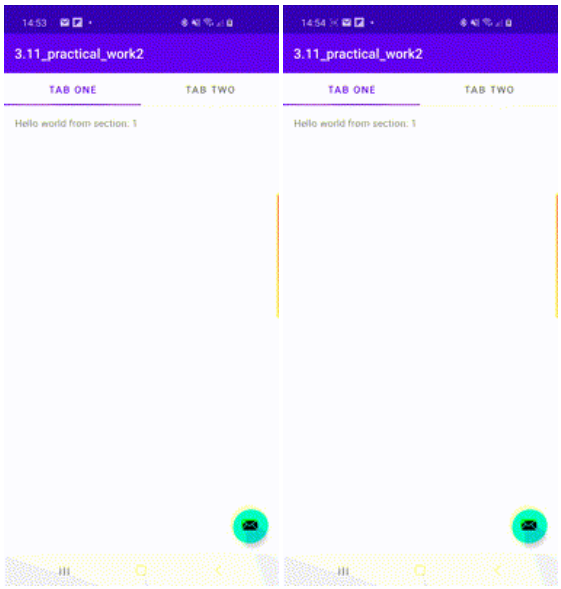
И самое существенное отличие состоит в том, что за переключение закладок теперь ответственен не сам viewPager а объект специального класса - TabLayoutMediator. Вот как теперь выглядит код активности:

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val viewPager: ViewPager2 = findViewById(R.id.view_pager)
        val tabs: TabLayout = findViewById(R.id.tabs)
        val sectionsPagerAdapter = SectionsPagerAdapter(this)
        viewPager.adapter = sectionsPagerAdapter

        TabLayoutMediator(tabs, viewPager,
            TabLayoutMediator.TabConfigurationStrategy { tab, position ->
                when (position) {
                    0 -> { tab.text = "TAB ONE"}
                    1 -> { tab.text = "TAB TWO"}
                }
            }).attach()
        // прочие действия в активности
         }
    }
}
```

Новый класс ViewPager2 содержит в себе много новых функций, например в нем можно легко изменить изменить направление скроллинга с горизонтального на вертикальный - достаточно просто добавить в макет атрибут orientation. Также в нем значительно больше подключаемых слушателей и т.д.



Начать тур для пользователя на этой странице