3.6. Фрагменты. Управление

Сайт: Samsung Innovation Campus
Курс: Мобильная разработка на Kotlin
Книга: 3.6. Фрагменты. Управление

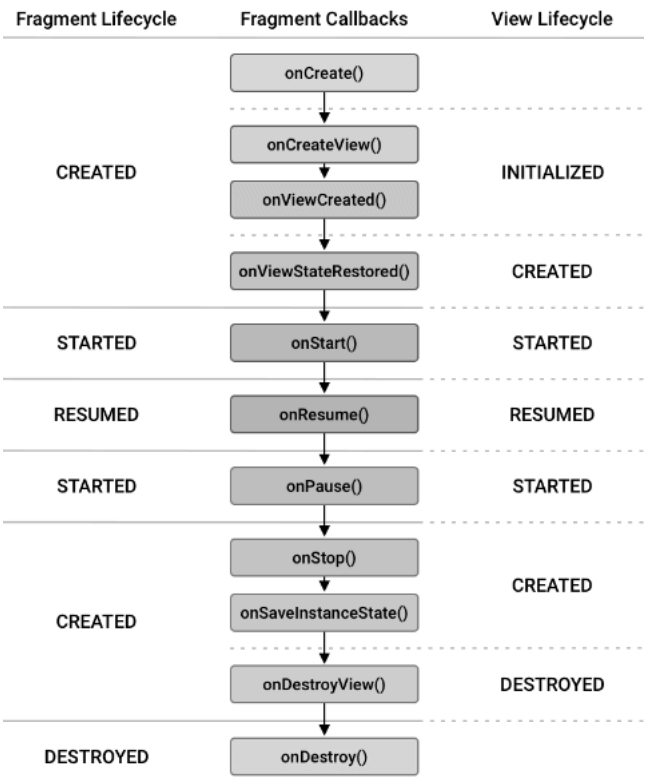
Напечатано:: Murad Rezvan
Дата: понедельник, 3 июня 2024, 17:50

Оглавление

Жизненный цикл фрагментов

Работа фрагментов также очень похожа на работу активностей и подчиняется своему жизненному циклу.

Перерисовать



- `onAttach()`: при выполнении этого метода фрагмент ассоциируется с определенной activity.
- `onCreate()`: происходит создание фрагмента.
- `onCreateView()`: фрагмент создает свой View. Обычно это делается при помощи функции `inflate` объекта `Inflater`, который передается в эту функцию.
- `onActivityCreated()`: вызывается после создания activity. С этого момента к компонентам интерфейса можно обращаться через метод findViewById()
- `onStart()`: вызывается, когда фрагмент становится видимым
- `onResume()`: фрагмент становится активным
- `onPause()`: фрагмент продолжает оставаться видимым, но уже не активен
- `onStop()`: фрагмент больше не является видимым
- `onDestroyView()`: уничтожается интерфейс, представляющий фрагмент
- `onDestroy()`: окончательно уничтожение фрагмента

Самыми важными, как и активностей, являются, пожалуй `onCreateView()`, выполняющая по сути роль функции `setContentView()` у активностей, `onResume()` в которой можно восстанавливать сохраненные значения и запускать фоновые процессы и `onPause()`, в которой можно сохранять значения и приостанавливать фоновые задачи.

То есть у фрагментов больше состояний, чем у активностей.

Коммуникация фрагментов и активности

Передача параметров при создании фрагментов

Создадим простейший фрагмент с `TextView`, в который будем передавать текст при создании.

Не рекомендуется это делать, добавляя конструктор с параметром.

Как при запуске активности мы можем добавить параметры в Bundle, так и при создании фрагмента мы можем создать Bundle-объект и передать его при помощи свойства `arguments` после создания:

```
val textFragment = TextFragment()
val bundle = Bundle()
bundle.putInt("text", )
textFragment.arguments = bundle
supportFragmentManager.beginTransaction()
        .add(R.id.fragment, textFragment)
```

а в функции `onCreateView()` класса фрагмента получить данные и использовать:

```
class TextFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
      return TextView(activity).apply {
          arguments?.getString("text", "")?.let{
              setText(it)
         }
      }
   }
}
```

При передаче параметров очень удобно использовать фабричную функцию `newInstance()`, которую Android Studio предлагает создать. Она выполнит всю работу по приему и распаковке параметров.

Для нашего класса TextFragment она может выглядеть так:

```
    companion object {
        @JvmStatic
        fun newInstance(text:String) =
            TestFragment().apply {
                arguments = Bundle().apply {
                    putString("text", text)
                }
            }
    }
```

С использованием фабричной функции создание фрагмента происходит в одну строку:

```
TextFragment fragment = TextFragment.newInstance("Some text")
```

При этом вероятность опечатки в названии параметра гораздо ниже.

Конечно, если фрагмент не будет принимать параметров, то особой пользы в фабричной функции нет.

Управление фрагментами из активности

К фрагментам активность может обращаться из своих же переменных, в которые она может сохранить ссылки на фрагменты при их создании.

Также есть метод supportFragmentManeger-а `findFragmentById()`, которому нужно передать id контейнера, например,

```
val fragment = findFragmentById(R.id.fragment_container);
```

Управление фрагментами из других фрагментов

Часто бывают ситуации, когда изменение в одном фрагменте приводит к необходимости изменению, например, замены другого. Такая коммуникация осуществляется через активность. То есть фрагменты передают команды по изменению других фрагментов активности, а она уже выполняет их.

Для этого у фрагмента есть свойство `activity`. В следующем разделе мы рассмотрим как можно элегантно это сделать при помощи интерфейса.

ListFragment

Списки очень часто используются в мобильных приложениях. Android Studio предлагает нам мастера, который сразу создает List Fragment. Создадим простой ListFragment из слов. Из выбранного слова будем формировать TestFragment, который мы разработали на предыдущем занятии.

Мастер сразу создает не только kt-файл, но и две разметки для самого списка и каждого его элемента. У нас будет всего одно слово, поэтому можно отказаться от разметки, используя стандартную разметку android `android.R.layout.simple_list_item_1`, либо исправить разметку элемента `fragment_word.xml`, оставив в ней один `textView`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="horizontal">

    <TextView
  android:id="@+id/content"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_margin="@dimen/text_margin"
  android:textAppearance="?attr/textAppearanceListItem" />

</LinearLayout>
```

Kotlin-код требует значительных правок.

1. Сначала поправим код самого фрагмента **`wordFragment.kt`**.

Во-первых, переделаем функцию `newInstance()` так, чтобы она принимала слово, а не количество колонок в макете:

```
companion object {
    @JvmStatic
  fun newInstance(word: String) =
        WordFragment().apply {
  arguments = Bundle().apply {
  putString("word", word)
           }
 }}
```

А в функции `onCreate()` распакуем это значение в переменную `word`

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    arguments?.getString("word")?.let{
        word = it
    }
 }
```

а строчку установки адаптера в функции `onCreateView` изменим так:

```
adapter = MyWordRecyclerViewAdapter(listener)
```

(Для наглядности среда создает целый пакет `dummy` с тестовыми данными, мы им пользоваться не будем. Лучше сразу удалить весь пакет и поправить те места, где используются классы из этого пакета на свои)

2. Далее исправим код адаптера **`MyWordRecyclerViewAdapter`**.

Именно этот класс регулирует все поведение списка.

Удалим из принимаемых параметров список из Dummy и создадим свой. Для краткости прямо в коде. Для нашего примера, возьмем список ключевых слов языка Kotlin:

```
private val mValues = listOf("package", "as", "typealias", "class", "this", "super", "val", "var", "fun", "for", "null", "true", "false",
"is", "in", "throw", "return", "break", "continue", "object", "if", "try", "else", "while", "do", "when", "interface", "yield", "typeof")
```

Во ViewHolder-е оставляем работу с одним полем:

```
inner class ViewHolder(val mView: View) : RecyclerView.ViewHolder(mView) {
    val mContentView: TextView = mView.content

    override fun toString(): String {
        return mContentView.text.toString()
    }
}
```

А в функции `onBindViewHolder()`

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val item = mValues[position]
    holder.mContentView.text = item

    with(holder.mView) {
  tag = item
        setOnClickListener(mOnClickListener)
    }
}
```

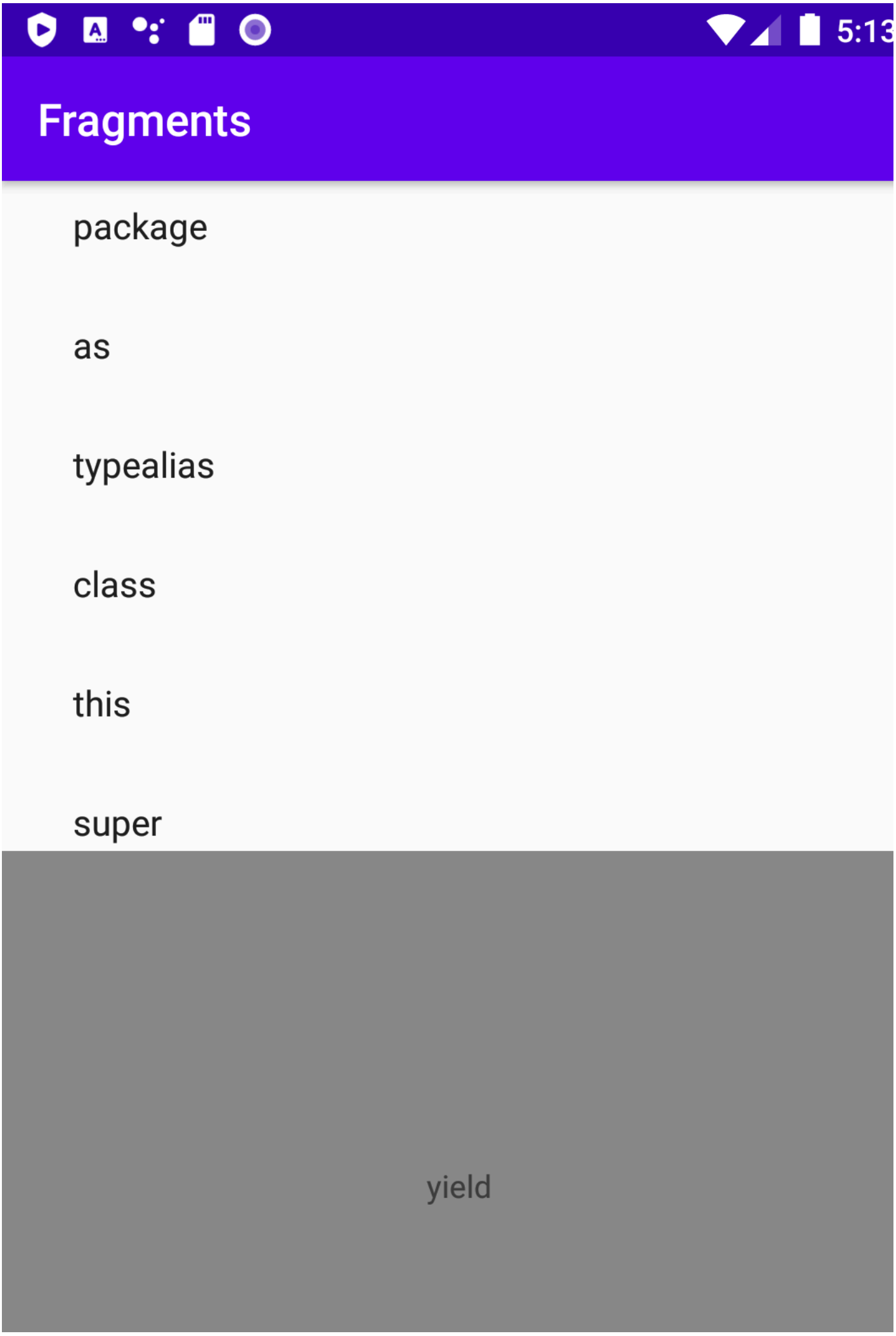
Если протестировать-запустить, программа упадет. Это от того, что сгенерированный код проверяет, установлен ли слушатель элементов, и если нет выбрасывает исключение в функции `onAttach()`

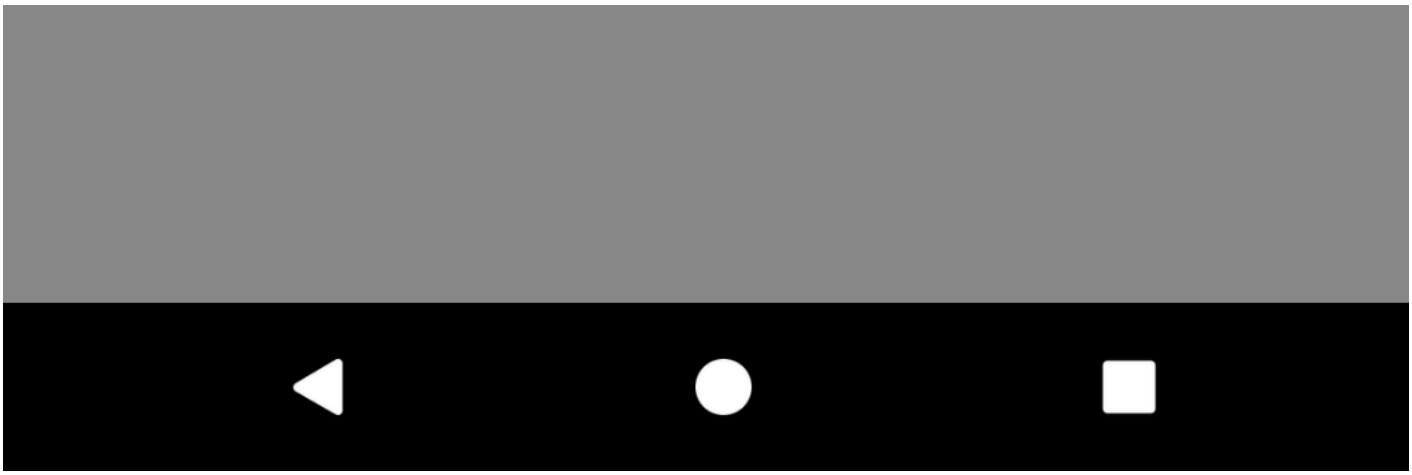
3. Остается поправить код активности, сделать ее слушателем:

```
class MainActivity : AppCompatActivity(), WordFragment.OnListFragmentInteractionListener{

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)
           if (savedInstanceState == null) {
               supportFragmentManager.beginTransaction()
               .add(R.id.fragment1, WordFragment())
                   .add(R.id.fragment2, TestFragment.newInstance(""))
                   .commit()
           }
        }

    override fun onListFragmentInteraction(item: String) {
        supportFragmentManager.beginTransaction()
            .replace(R.id.fragment2, TestFragment.newInstance(item))
            .commit()
    }
}
```


5:13
Fragments
package
as
typealias
class
this
super
yield



Начать тур для пользователя на этой странице