

ЛЕКЦИЯ I © 2020 Нет Ит

Android: Fragments

Теодор Костадинов



SOFTWARE
ACADEMY



Съдържание

1. Context
2. Какво представляват фрагментите
3. fragment lifecycle
4. добавяне на фрагменти
5. комуникация с activity-то
6. Navigation Component
7. NavHost
8. NavController
9. Домашно

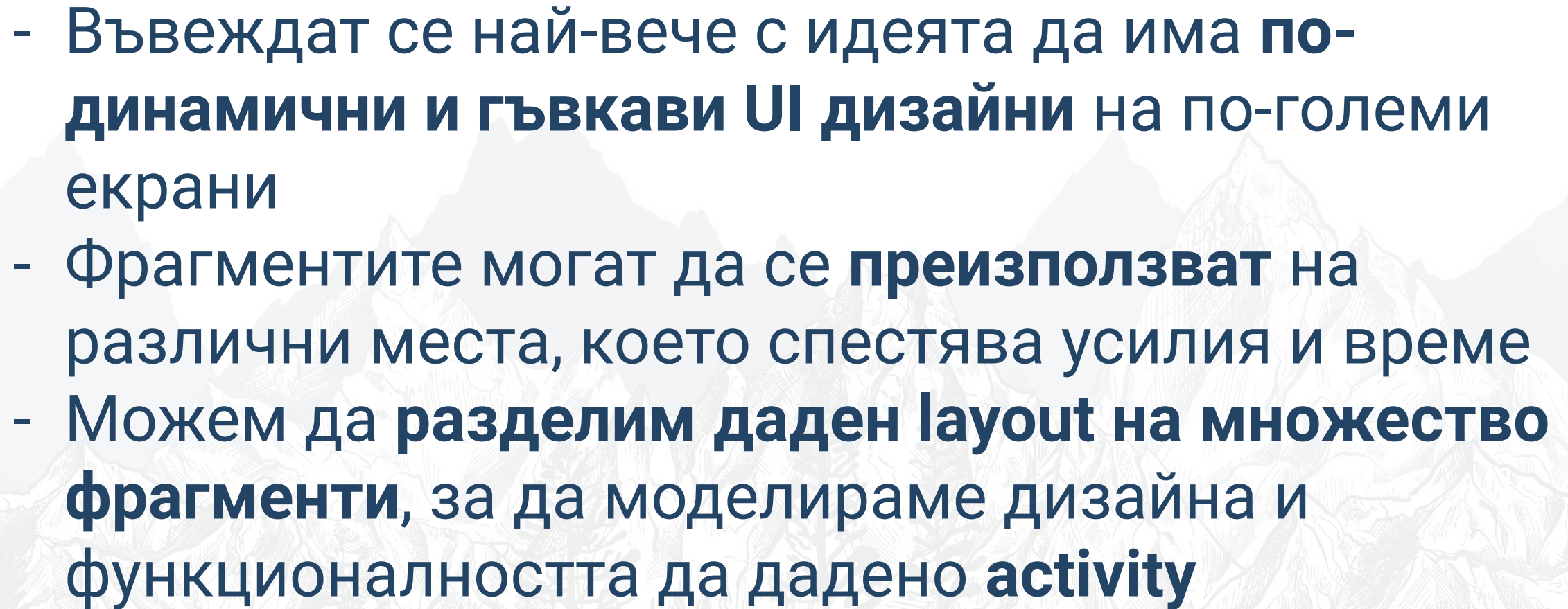


CONTEXT

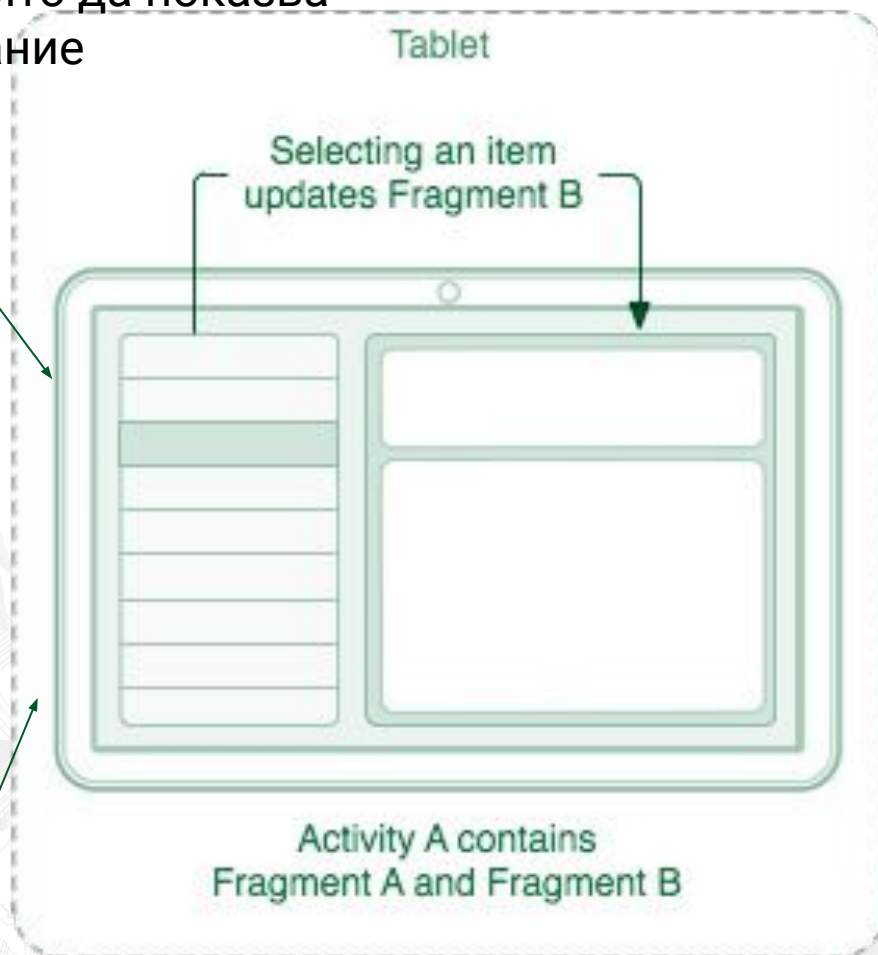
- Обект, който съхранява инфо за текущото състояние на апп-а
 - Кой е апп-а
 - Кой екран в момента е активен
 - Къде се намира апп-а в паметта на телефона
 - Къде се намират ресурсите на апп-а (картинки, медия, файлове и тн)
- Всяко активити наследява контекст
- Всяко вю, което е в работещо активити има връзка към контекста



FRAGMENTS

- 
- Въвеждат се най-вече с идеята да има **подинамични и гъвкави UI дизайни** на по-големи екрани
 - Фрагментите могат да се **преизползват** на различни места, което спестява усилия и време
 - Можем да **разделим даден layout на множество фрагменти**, за да моделираме дизайна и функционалността да дадено **activity**

Използваме един фрагмент, който да показва името на дадена статия, и друг, който да показва нейното съдържание



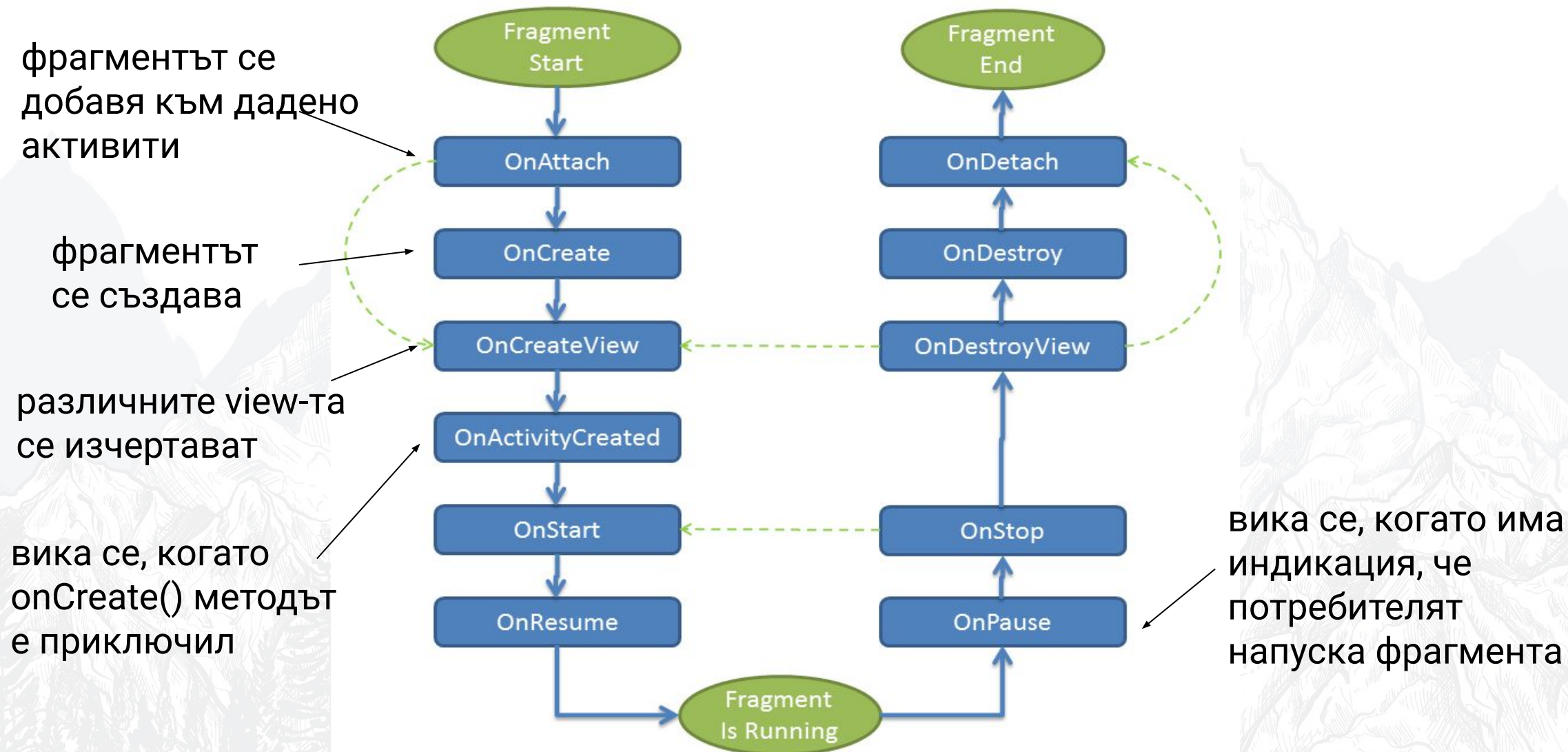
Тъй като екранът е голям, двата фрагмента се показват едновременно

Преизползваме същите фрагменти, но на по-малки устройства се показват едно след друго



- Могат да бъдат разглеждани като подактививита, въпреки че в действителност са **прости ViewGroup-и**
- Фрагментите, както и **activity-тата**, имат **lifecycle**
- Едно **activity** може да има много фрагменти, които да управлява
- Фрагментите могат да бъдат закачани, откачани, променяни, докато даденото **activity** работи
- Животът на даден фрагмент се определя от управляващото го **activity**, но обратното не е вярно

Lifecycle



- **Lifecycle**-ът на фрагментите е много сходен с този на активититата
- Най-голяма разлика има при `onCreateView` метода, който се грижи за **inflate**-ването на **layout-a на дадения фрагмент** (XML файла)
- Важно е да запомните, че фрагментите имат задължително **празен конструктор**

Два начина да се добавят фрагменти:

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <fragment android:name="com.example.news.ArticleListFragment"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" />  
    </LinearLayout>
```

Добавяме го **статично**,
но така губим
възможността да го
преизползваме

Използваме tag-ът **fragment**
директно в XML-а на нашето
activity

Два начина да се добавят фрагменти:

Динамично с помощта на FragmentManager

```
FragmentManager fragmentManager = getSupportFragmentManager();  
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
ExampleFragment fragment = new ExampleFragment();  
fragmentTransaction.add(R.id.fragment_container, fragment);  
fragmentTransaction.commit();
```

Трябва да кажем **кой фрагмент** искаме да заредим и да окажем **контейнер**, в който да се разположи

- Всички промени, свързани с дадено **activity** и фрагментите, които управлява, се наричат **транзакции**
- За да можем да правим такива промени е нужно да имаме инстанция на `FragmentManager`
- **`add()`**, **`remove()`**, **`replace()`**
- За да приключи дадена транзакция, викаме **`commit()`**

ПОЧИВКА
до 20:30



addToBackStack()

```
Fragment newFragment = new ExampleFragment();
FragmentManager transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

Ако искаме потребителят да може да се връща към **предишното състояние на фрагмента**, преди изпълнението на commit, можем да добавим транзакцията към back stack-а. **Back stack-а се управлява от activity-то.**

Как даден фрагмент си комуникира с activity-то:

```
public static class FragmentA extends Fragment {  
    ...  
    // Container Activity must implement this interface  
    public interface OnArticleSelectedListener {  
        public void onArticleSelected(Uri articleUri);  
    }  
    ...  
}
```

За да може да си комуникира с activity-то, ще създадем **интерфейс**, който да следи за това. Ще задължим **activity-то да го имплементира** и да зададе нужното поведение.

Как даден фрагмент си комуникира с activity-то:

```
public static class FragmentA extends ListFragment {  
    OnArticleSelectedListener listener;  
  
    ...  
  
    @Override  
    public void onAttach(Context context) {  
        super.onAttach(context);  
        try {  
            listener = (OnArticleSelectedListener) context;  
        } catch (ClassCastException e) {  
            throw new ClassCastException(context.toString() +  
                " must implement OnArticleSelectedListener");  
        }  
    }  
    ...  
}
```

За да сме сигурни, че activity-то ни наистина имплементира нужната логика, служеща за комуникация, в **onAttach()** метода правим нужната проверка.

ВЪПРОСИ?



© 2020 Нет Ит

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!



SOFTWARE
ACADEMY

