<TeachMeSkills/>

Школа программирования
teachmeskills.com

курс

Android разработчик

Занятие 15. Введение в Android. Часть 3

Агенда занятия

Intent

Fragment. Жизненный цикл

Fragment Manager

NavController, NavGraph

Передача данных между фрагментами. Разные способы

▶ Intent

Fragment. Жизненный цикл

Fragment Manager

NavController, NavGraph

Передача данных между фрагментами. Разные способы

Intent

Intent (намерение) — механизм для описания одной операции — выбор фотографии, отправка письма, звонок, запуск браузера и так далее. Наиболее распространенный сценарий использования — запуск другой активити в своём приложении.

Помимо этого, используется для запусков сервисов.

Существует 2 вида:

- явное
- неявное

Intent. Явный Intent

Ключевая особенность явного намерения — прямое указание имени класса.

```
private fun openSecondActivity() {
    // Явное указание класса Activity, которое нужно открыть
    val intent = Intent( packageContext: this, SecondActivity::class.java)
    startActivity(intent)
}
```

Intent. Неявный Intent

Отличие от явного намерения в том, что мы заранее не знаем, что именно будет открыто в последствии вызова нашего намерения. Система будет искать активити, которая подходит по указанным параметрам. Если их будет несколько, пользователю будет предоставлен выбор.

```
private fun openBrowser() {
    findViewById<Button>(R.id.button).setOnClickListener {
        // Неявный интент для открытия веб-страницы
        val intent = Intent(Intent.ACTION_VIEW, Uri.parse( uriString: "https://google.com"))
        startActivity(intent)
    }
}
```

Intent. Неявный Intent

Отличие от явного намерения в том, что мы заранее не знаем, что именно будет открыто в последствии вызова нашего намерения. Система будет искать активити, которая подходит по указанным параметрам. Если их будет несколько, пользователю будет предоставлен выбор.

```
private fun openBrowser() {
    findViewById<Button>(R.id.button).setOnClickListener {
        // Неявный интент для открытия веб-страницы
        val intent = Intent(Intent.ACTION_VIEW, Uri.parse( uriString: "https://google.com"))
        startActivity(intent)
    }
}
```

Intent. Неявный Intent

```
        findViewById<Button>(R.id.button).setOnClickListener {
            // Неявный интент для открытия телефона
            val intent = Intent(Intent.ACTION_DIAL, Uri.parse( uriString: "tel:+123456789"))
            startActivity(intent)
        }
    }
```

```
        findViewById<Button>(R.id.button).setOnClickListener {
            // Неявный интент для открытия камеры
            val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            startActivity(intent)
        }
    }
```

Intent. Неявный Intent

```
        findViewById<Button>(R.id.button).setOnClickListener {
            // Неявный интент для открытия камеры
            val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            // Проверка наличия приложения для обработки интента
            if (intent.resolveActivity(packageManager) != null) {
                startActivity(intent)
            } else {
                Toast.makeText( context: this,  text: "There is no camera app installed", Toast.LENGTH_SHORT).show()
            }
        }
    }
```

Intent. Передача данных

Для передачи данных через Intent используется метода getExtra(), для получения get(Тип_данных)Extra(). Данные хранятся в паре ключ-значение

Intent. Передача данных

```kotlin
class MainActivity : AppCompatActivity() {

    private var editText: EditText? = null
    private var acceptButton: Button? = null


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
        editText = findViewById(R.id.editText)
        acceptButton = findViewById(R.id.button)
        openBrowser()
    }

    private fun openBrowser() {
        acceptButton?.setOnClickListener {
            val intent = Intent( packageContext: this, SecondActivity::class.java)
            val userName = editText?.text.toString()
            // Передача строки из EditText во вторую активность
            intent.putExtra(USER_NAME_KEY, userName)
            startActivity(intent)
        }
    }

    companion object {
        const val USER_NAME_KEY = "USER_NAME"
    }

}
```

Intent. Передача данных

```
class SecondActivity : AppCompatActivity() {

    private var userTextView: TextView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
        userTextView = findViewById(R.id.userNameTextView)
        setUsername()
    }

    private fun setUsername() {
        // Получение строки из первой активности
        val userName = intent.getStringExtra(MainActivity.USER_NAME_KEY)
        userTextView?.text = userName
    }
}
```

Intent

- **Fragment. Жизненный цикл**

Fragment Manager

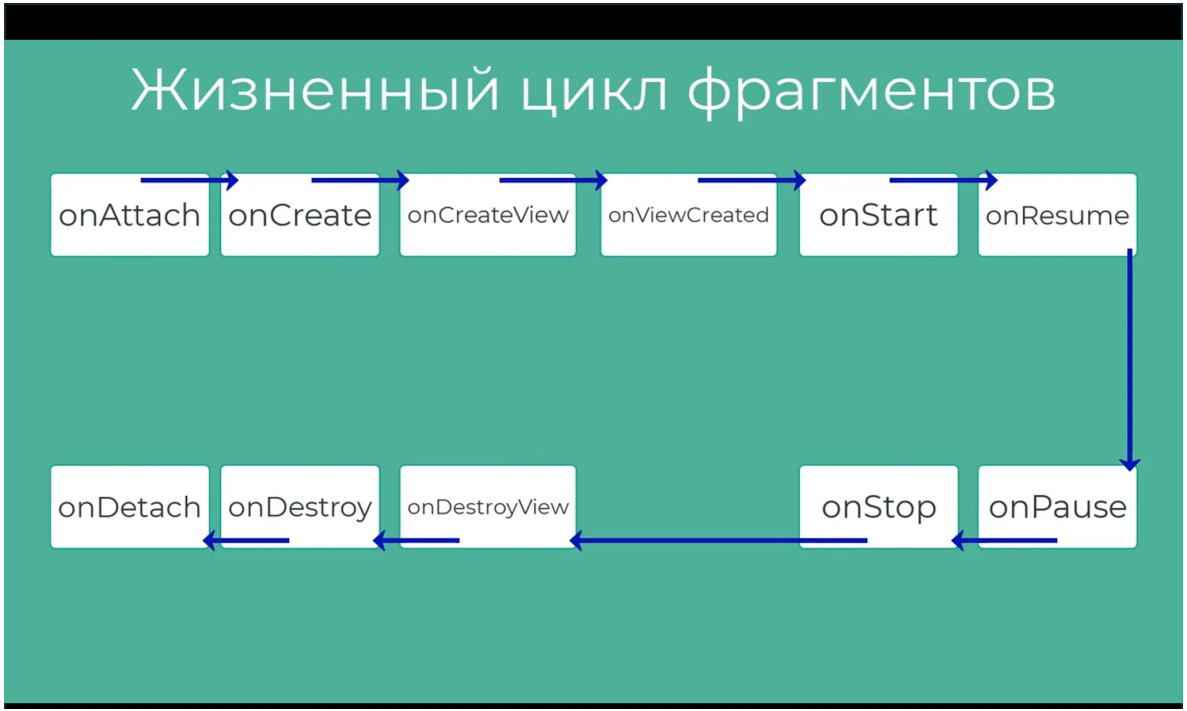
NavController, NavGraph

Передача данных между фрагментами. Разные способы

Fragment

Фрагмент (Fragment) – это часть пользовательского интерфейса, которая работает внутри Activity. Он помогает разделять логику и переиспользовать UI в разных экранах.

Fragment


Жизненный цикл фрагментов
onAttach
onCreate
onCreateView
onViewCreated
onStart
onResume
onPause
onStop
onDestroyView
onDestroy
onDetach

Fragment

Фрагмент не является подклассом Context, поэтому для доступа к конексту используется ***activity/requireActivity()/requireContext().***

Fragment

```
<androidx.fragment.app.FragmentContainerView
    android:name="com.example.myfirstapp.FirstFragment"
    android:id="@+id/fragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Intent

Fragment. Жизненный цикл

▶ **Fragment Manager**

NavController, NavGraph

Передача данных между фрагментами. Разные способы

FragmentManager

FragmentManager — это основной инструмент для работы с фрагментами в Android. Он управляет их добавлением, удалением, заменой и состоянием в рамках Activity или Fragment. В конце любой цепочки транзакции необходимо вызывать метод ***commit()****.*

```
    // Замена фрагмента
    supportFragmentManager.beginTransaction()
        // Заменить фрагмент в контейнере на FirstFragment
        .replace(R.id.fragmentContainer, FirstFragment())
        .commit()
}
```

FragmentManager. Методы транзакции

- add()— Добавляет фрагмент к активности
- remove() — Удаляет фрагмент из активности
- replace() —Заменяет один фрагмент на другой
- hide() — Прячет фрагмент (делает невидимым на экране)
- show() — Выводит скрытый фрагмент на экран
- detach() (API 13) — Отсоединяет фрагмент от графического интерфейса, но экземпляр класса сохраняется
- attach() (API 13) — Присоединяет фрагмент, который был отсоединён методом detach()

FragmentManager. getInstance()

Довольно часто при переходе на новый фрагмент, ему необходимо передать какие-либо данные. Для таких случаев существует паттерн создания статического метода внутри фрагмента, который возвращает его экземпляр с bundle внутри.

```
companion object {

    fun getInstance(username: String): SecondFragment {
        return SecondFragment().apply {
            arguments = Bundle().apply {
                putString(USER_NAME_KEY, username)
            }
        }
    }
}
```

FragmentManager. getInstance()

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val username = arguments?.getString(USER_NAME_KEY) ?: ""
    view.findViewById<TextView>(R.id.userNameTextView).text = username
}
```

FragmentManager. BackStack

Фрагменты, как и активити, могут управляться системной кнопкой назад. Для этого необходимо добавить фрагмент в стек и после нажатия на системную кнопку назад, вы вернетесь на этот фрагмент.

Если вы вызовете метод addToBackStack() при удалении или замещении фрагмента, то будут вызваны методы фрагмента onPause(), onStop(), onDestroyView().

Когда пользователь нажимает на кнопку возврата, то вызываются методы фрагмента onCreateView(), onActivityCreated(), onStart() и onResume().

FragmentManager. BackStack

```
        requireActivity().supportFragmentManager.beginTransaction()
            .replace(R.id.fragmentContainer, secondFragment)
            // Добавление фрагмента в backstack
            .addToBackStack( name: null)
            .commit()
    }
}
```

Intent

Fragment. Жизненный цикл

Fragment Manager

- **NavController, NavGraph**

  Передача данных между фрагментами. Разные способы

JetPack navigation

Библиотека, созданная компанией JetPack для упрощения навигацией между экранами. Состоит из двух главных компонентов — NavGraph — граф навигации, описывающий навигацию экранами и NavController — класс, управляющий переходами.

Зависимости:

implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"

implementation "androidx.navigation:navigation-ui-ktx:$nav_version"

JetPack navigation. NavGraph

NavGraph — это XML-файл, в котором описаны все экраны (дестинации) и связи между ними внутри приложения.

```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/nav_graph"
    app:startDestination="@id/firstFragment">

    <fragment
        android:id="@+id/firstFragment"
        android:name="com.example.myfirstapp.FirstFragment"
        android:label="fragment_first"
        tools:layout="@layout/fragment_first" >
        <action
            android:id="@+id/action_firstFragment_to_secondFragment"
            app:destination="@id/secondFragment" />
    </fragment>

    <fragment
        android:id="@+id/secondFragment"
        android:name="com.example.myfirstapp.SecondFragment"
        android:label="fragment_second"
        tools:layout="@layout/fragment_second" />
</navigation>
```

JetPack navigation. NavGraph

Создает в отдельной директории /res/navigation/nav_graph.xml

JetPack navigation. NavController

*`NavController` — это основной компонент **Jetpack Navigation**, который управляет навигацией между фрагментами в приложении. Он отвечает за:*

- *Переходы между экранами (фрагментами)*
- *Обработку кнопки "Назад"*
- *Управление стеком навигации*

JetPack navigation. NavController

NavController привязан к NavHostFragment, который работает как контейнер для навигации.

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainer"
    android:name="androidx.navigation.fragment.NavHostFragment"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

JetPack navigation. NavController

Инициализация NavController в активити:

```
navController = Navigation.findNavController( activity: this, R.id.fragmentContainer)
navController = findNavController(R.id.fragmentContainer)
```

JetPack navigation. NavController

Инициализация NavController во фрагменте:

```
var navController = Navigation.findNavController(view)
navController = findNavController()
```

JetPack navigation. NavController. Методы

- *navController.navigate(R.id.action_homeFragment_to_detailsFragment) — транзакция*
- *navController.popBackStack() — возвращение на предыдущий экран*
- *navController.popBackStack(R.id.homeFragment, false) — возврат на определенный фрагмент с очисткой стека*
- *navController.navigate(R.id.action_homeFragment_to_detailsFragment, bundle) — транзакция с передачей параметров*

Intent

Fragment. Жизненный цикл

Fragment Manager

NavController, NavGraph

- **Передача данных между фрагментами. Разные способы**

Передача данных между фрагментами

- Bundle
- navArguments
- Интерфейс (лучше не использовать)
- FragmentResult

NavArguments

Для доступа, нужно добавить плагин androidx.navigation:navigation-safe-args-gradle-plugin:$version.

После добавления плагина, откроется возможность добавлять аргументы напрямую в ваш путь, создавая конкретный класс.

NavArguments

Добавление аргумента:

```
<fragment
    android:id="@+id/secondFragment"
    android:name="com.example.mysecondapp.SecondFragment"
    android:label="fragment_second"
    tools:layout="@layout/fragment_second" >
    <action
        android:id="@+id/action_secondFragment_to_thirdFragment"
        app:destination="@id/thirdFragment" />

    <argument
        android:name="text"
        app:argType="string" />
</fragment>
```

NavArguments

Передача аргумента в момент навигации:

```
private fun navigateToSecondFragment(text: String) {
    navController?.navigate(FirstFragmentDirections.actionFirstFragmentToSecondFragment(text))
}
```

NavArguments

Получение класса, содержащего аргументы:

```
private val arguments: SecondFragmentArgs by navArgs()
```

FragmentResult

*Метод передачи данных между фрагментами через **fragmentManager**. Фрагмент, который передает данные, вызывает метод **setFragmentResult(key, bundle)**, где key — ключ, по которому, другой фрагмент будет искать бандл, а bundle — бандл с данными. Фрагмент, получающий данные, вызывает метод **setFragmentResultListener(requestKey, lifeCycleOwner, listener)**, где requestKey — ключ, lifecycleOwner — сам фрагмент и listener — реализация слушателя.*

FragmentResult

```
parentFragmentManager.setFragmentResult( requestKey: "Key", bundleOf( ...pairs: USER_NAME_KEY to text))
```

```
private fun listenFragmentResult() {
    parentFragmentManager.setFragmentResultListener(
        requestKey: "Key",
        lifecycleOwner: this
    ) { requestKey, bundle ->
        val result = bundle.getString(USER_NAME_KEY)
        Log.d( tag: "FragmentResult",  msg: "Received result: $result")
    }
}
```

Задачи

Задача 1: Запуск новой Activity с передачей данных через Intent

Открыть SecondActivity из MainActivity, передав строку "Hello, Second Activity!".

Задачи

Задача 2: Добавление фрагмента в контейнер (FragmentManager)

Задание: Добавь MyFragment в контейнер FrameLayout.

Задачи

Задача 3: Замена одного фрагмента на другой (replace)
Задание: В MainActivity есть два фрагмента (FragmentA, FragmentB). При нажатии на кнопку заменяй FragmentA на FragmentB.

Задачи

Задача 4: Передача данных между фрагментами (Bundle)

Задание: Передай строку "Data from Activity" из Activity в Fragment.

Q&A

Домашнее задание

Задачи

Задача 1: Переход через несколько фрагментов с сохранением состояний

Шаги:

1. Создай несколько фрагментов, например, HomeFragment, DetailsFragment, и SettingsFragment.
2. В HomeFragment размести кнопку для перехода в DetailsFragment, а из DetailsFragment — в SettingsFragment.
3. Для каждого фрагмента используй NavGraph с переходами.
4. Убедись, что при возвращении на предыдущие фрагменты их состояния сохраняются.

Подсказка: Используй addToBackStack для сохранения состояний при переходах.

Задачи

Задача 2: Переход между фрагментами с передачей аргументов

Шаги:

1. Создай два фрагмента: FirstFragment и SecondFragment.
2. В FirstFragment создавай кнопку, по нажатию на которую будет передаваться строка в SecondFragment.
3. В SecondFragment отобрази переданное сообщение.
4. Используй Safe Args для передачи данных через NavGraph.

Подсказка: В nav_graph.xml нужно указать аргумент для SecondFragment.

Задачи

Задача 3: Обработка возвращаемых данных из фрагмента

Шаги:

1. Создай два фрагмента: FirstFragment и SecondFragment.
2. На экране FirstFragment размести кнопку, при нажатии на которую откроется SecondFragment.
3. В SecondFragment создай кнопку, которая при нажатии отправляет результат обратно в FirstFragment через setResult() и NavController.
4. В FirstFragment отобрази возвращённые данные в TextView.

Подсказка: Используй setResult() для передачи данных назад.

Q&A

Ваши вопросы

Спасибо