المحاضرة التاسعة:

التعامل مع Fragment

مدرسو المقرر:

م. محمود الجلخ.

م. رهام العمر.

م. هلا بريمان.

مقدمة:

في عالم تطوير تطبيقات Android، يعتبر مفهوم Fragments من المفاهيم المحورية التي تتيح للمطورين بناء واجهات مستخدم مرنة وقابلة لإعادة الاستخدام(ديناميكية). تم تقديم Fragments لأول مرة في Android 3.0 لدعم الشاشات الكبيرة مثل الأجهزة اللوحية، لكنها أصبحت لاحقاً جزءاً أساسياً من تطوير التطبيقات الحديثة. في هذا البحث، سنناقش بتفصيل مفهوم الـ Fragment، كيفية إنشائه، دورة حياته، والتواصل بينه وبين الـ Activity، مع تضمين أمثلة برمجية توضيحية.

ما هو الـ Fragment؟

الـ Fragment هو جزء قابل لإعادة الاستخدام من واجهة المستخدم يمكن دمجه داخل Activity. يمكن استخدام عدة Fragments داخل نفس الـ Activity لعرض واجهات متعددة. يسمح ال Fragment بإدارة مستقلة لواجهة المستخدم الخاصة به والتفاعل مع المستخدم، وكل قسم يمكن أن يعرض واجهة مختلفة أو معلومات معينة، ويتم التحكم بها بشكل مرن أثناء تشغيل التطبيق ويكون دائماً مرتبطاً بـ Activity مضيف.

لماذا نستخدم Fragments؟

تم تصميم Fragments لمعالجة عدة تحديات في تطوير التطبيقات مثل:

1. الحاجة إلى إعادة استخدام الواجهات في أكثر من مكان.
2. دعم التكيّف مع الشاشات الكبيرة مثل الأجهزة اللوحية.
3. تقليل تعقيد إدارة الأنشطة (Activities) والتنقل بينها.
4. تحسين الأداء من خلال استبدال أجزاء من الشاشة بدلاً من تحميل شاشة جديدة بالكامل.

إنشاء Fragment :

لإنشاء Fragment، يجب إنشاء كلاس يرث من Fragment وتحديد واجهة المستخدم الخاصة به باستخدام ملف XML. يمكن إضافته إما من خلال ملف XML للـ Activity أو بشكل برمجي.

مثال برمجي:

```
class MyFragment extends Fragment {
    public MyFragment() {
        super(R.layout.fragment_my);
    }
}
```

إضافة Fragment برمجياً داخل الـ Activity:

```
FragmentManager fragmentManager = getSupportFragmentManager();
fragmentManager.beginTransaction()
    .setReorderingAllowed(true)
    .add(R.id.fragment_container_view, MyFragment.class, null)
    .commit();
```

إدارة الـ Fragments باستخدام FragmentManager :

يتم استخدام FragmentManager لتنفيذ العمليات على الـ Fragments مثل الإضافة والاستبدال والحذف. كما يمكن حفظ الحالة في الـ Back Stack لتوفير تجربة تنقل سلسة للمستخدم. العمليات تُدار باستخدام FragmentTransaction.

دورة حياة الـ Fragment :

لكل Fragment دورة حياة خاصة به، تتضمن الحالات: INITIALIZED، CREATED، STARTED، RESUMED، DESTROYED. ويتم استخدام توابع مثل:

- **Initial State:**
  - FragmentA: onAttach → onCreate → onCreateView → onViewCreated → onStart → onResume.
- **Navigate to FragmentB:**
  - FragmentA: onPause → onStop → onDestroyView.
  - FragmentB: onAttach → onCreate → onCreateView → onViewCreated → onStart → onResume.
- **Back Press:**
  - FragmentB: onPause → onStop → onDestroyView → onDestroy → onDetach.
  - FragmentA: onCreateView → onViewCreated → onStart → onResume.

والـ FragmentManger لـه onAttach()&onDeatch()

وغيرها للتعامل مع كل مرحلة من مراحل الحياة.

التواصل بين Activity و Fragment :

يتم تمرير البيانات من Activity إلى Fragment باستخدام Bundles وnewInstance()، ومن Fragment إلى Activity أو بين Fragments باستخدام ViewModel أو FragmentResult API لضمان تواصل مرن وآمن.

مثال شامل لتطبيق ال Fragment

الخطوة الأولى: إنشاء كلاس يرث من Fragment:

```
class MyFragment extends Fragment {
    public MyFragment() {
        super(R.layout.fragment_my);
    }
}
```

الخطوة الثانية: إنشاء الواجهة الخاصة به باستخدام XML:

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_blue_light">

    <TextView
        android:id="@+id/txtMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="مرحباً من Fragment"
        android:textSize="20sp"
        android:layout_gravity="center" />
</FrameLayout>
```

الخطوة الثالثة: إضافته إلى Activity إما باستخدام XML أو برمجياً:

1. باستخدام XML:

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

2. برمجياً داخل onCreate في MainActivity:

```
FragmentManager fragmentManager = getSupportFragmentManager();
fragmentManager.beginTransaction()
    .setReorderingAllowed(true)
    .add(R.id.fragment_container_view, MyFragment.class, null)
    .commit();
```

الكود :

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(v -> {
            FragmentManager fragmentManager =
getSupportFragmentManager();
            fragmentManager.beginTransaction()
                .setReorderingAllowed(true)
                .add(R.id.fragment_container_view,
ExampleFragment.class, null)
                .commit();
        });
    }
}
```

activity_main.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
<Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="إضافة Fragment" />
<androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragment_container_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

ExampleFragment.java

```
public class ExampleFragment extends Fragment {
    private static final String ARG_PARAM1 = "param1";
    private String mParam1;

    public static ExampleFragment newInstance(String param1) {
        ExampleFragment fragment = new ExampleFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_example,
container, false);
        TextView textView = view.findViewById(R.id.txtMessage);
        textView.setText(mParam1);
        return view;
    }
}
```

fragment_example.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#BBDEFB">

    <TextView
        android:id="@+id/txtMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:text="النص هنا"
        android:textSize="22sp" />
</FrameLayout>
```

خاتمة

يُعتبر استخدام Fragments أداة فعّالة في تطوير التطبيقات الحديثة، حيث يوفر قابلية لإعادة الاستخدام وتحكم أفضل بواجهة المستخدم، خاصة عند دعم تعدد الشاشات.

إن فهم كيفية التعامل مع دورة الحياة والتواصل بين المكونات يوفر للتطبيق استقراراً وأداء عالٍ وتجربة مستخدم متميزة.

انتهت المحاضرة

بالتمنيات بالتوفيق الدائم لجميع طلبتنا الأكارم