IT2010 – Mobile Application Development
BSc (Hons) in Information Technology
2nd Year
Faculty of Computing
SLIIT

2023 – Lab 03

Lab Sheet: Android Intents

Introduction

Intents are a fundamental building block in Android development. They enable components (such as activities) to request functionality from other components of the Android system. Intents can be explicit (targeting a specified component) or implicit (targeting any component that can handle the action and data).

In this lab, you will learn:

1. The basics of using intents to start activities.
2. How to pass data between activities using intents.
3. Using implicit intents to leverage other apps' functionalities.

Starting a New Activity using Intent

Learn how to navigate from one activity to another using an explicit intent.

Steps:

1. **Create a New Android Project** in Android Studio.
2. Create two activities: **MainActivity** and **SecondActivity**.
3. In **MainActivity**, place a Button labeled "Go to Second Activity".
4. Add an **OnClickListener** to the button. Inside the listener, use an explicit intent to start **SecondActivity**.

```
val btnNavigate: Button = findViewById(R.id.btnNavigate)
btnNavigate.setOnClickListener {
   val intent = Intent(this, SecondActivity::class.java)
   startActivity(intent)
}
```

Task 2: Passing Data using Intents

Learn how to send data from one activity to another.

Steps:

1. In **MainActivity**, add an EditText widget for the user to enter their name.
2. Modify the button's **OnClickListener** to retrieve the name from the EditText and put it as an extra into the intent.
3. In **SecondActivity**, retrieve the name from the intent and display it in a TextView

```
// MainActivity.kt
val btnSendData: Button = findViewById(R.id.btnSendData)
btnSendData.setOnClickListener {
   val name = editTextName.text.toString()
   val intent = Intent(this, SecondActivity::class.java)
   intent.putExtra("USER_NAME", name)
   startActivity(intent)
}

// SecondActivity.kt
val intent = intent
val name = intent.getStringExtra("USER_NAME")
textViewGreeting.text = "Hello, $name!"
```

Task 3: Using Implicit Intents

Objective:

Leverage other apps to perform an action, e.g., opening a web page in a browser.

Steps:

1. In **MainActivity**, place another Button labeled "Open Web Page".
2. Set an **OnClickListener** for the button to open a URL in a browser using an implicit intent.

```
val btnOpenWeb: Button = findViewById(R.id.btnOpenWeb)
btnOpenWeb.setOnClickListener {
   val webpage = Uri.parse("http://www.google.com")
   val intent = Intent(Intent.ACTION_VIEW, webpage)
   if (intent.resolveActivity(packageManager) != null) {
      startActivity(intent)
   }
}
```

Phone Dialer

In your MainActivity, you can add a button labeled "Dial Phone" and then set an OnClickListener for the button to open the phone dialer with a predefined number.

```
val btnDialPhone: Button = findViewById(R.id.btnDialPhone)
btnDialPhone.setOnClickListener {
   val phoneUri = Uri.parse("tel:1234567890")
   val intent = Intent(Intent.ACTION_DIAL, phoneUri)
   if (intent.resolveActivity(packageManager) != null) {
      startActivity(intent)
   }
}
```

Open Camera

Use an implicit intent to capture a photo using the device's camera app. Note that this is only available after API level 23.

1. Implement the necessary permissions in the 'AndroidManifest.xml'

```
<uses-feature android:name="android.hardware.camera" android:required="true" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

2. In your MainActivity, add a button labeled "Capture Photo" and then set an OnClickListener for the button:

```
val btnCapturePhoto: Button = findViewById(R.id.btnCapturePhoto)
btnCapturePhoto.setOnClickListener {
   val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
   if (intent.resolveActivity(packageManager) != null) {
      startActivity(intent)
   }
}
```

Custom Filters

Create a custom intent filter for your SecondActivity, so it can be launched by implicit intents from other apps.

Update the AndroidManifest.xml to add an intent filter to SecondActivity:

```
<activity android:name=".SecondActivity">
   <intent-filter>
      <action android:name="com.example.MY_CUSTOM_ACTION" />
      <category android:name="android.intent.category.DEFAULT" />
   </intent-filter>
</activity>
```

From any other activity or even a different app, you can now launch SecondActivity using the custom action:

```
val intent = Intent("com.example.MY_CUSTOM_ACTION")
if (intent.resolveActivity(packageManager) != null) {
   startActivity(intent)
}
```