Mobile Applications CSCI 448 Lecture 23

Completing the Second Activity

Feedback Loop

Learning Outcomes For Today

- Create an app that uses multiple activities and send data between activities
- Discuss the OS role in the activity life cycle
- Explain the process to receive a response from a second activity

On Tap For Today

- Starting A Second Activity

- Passing Data To Activities
- Returning Data From Activities

- Practice

On Tap For Today

- Starting A Second Activity

- Passing Data To Activities
- Returning Data From Activities

- Practice

Explicit Intent

CallingActivity.kt

```
val i = TargetActivity.newIntent(context, extraValue)

startActivity( i )
```

TargetActivity.kt

```
companion object {
  private const val EXTRA_KEY = "key"
  fun newIntent(pkgCtxt: Context, value: String): Intent {
    val i = Intent( pkgCtxt, TargetActivity::class.java )
    i.putExtra( EXTRA_KEY, value )
    return i
  }
}

override fun onCreate(savedInstanceState: Bundle?) {
  val data = intent.getStringExtra(EXTRA_KEY, "")
}
```

Implicit Intent

```
onClickListener = {
        val str = "string value"

        val i = Intent(Intent.ACTION_SEND).apply {
          type = "text/plain"
          putExtra(Intent.EXTRA_TEXT, str)
        }
        startActivity(i)
}
```

Start the Activity

- Launching new Activities
  - When no response needed:

    `startActivity(intent)`

On Tap For Today

- Starting A Second Activity
- Passing Data To Activities
- Returning Data From Activities
- Practice

Sending Data to Target Activity

```
val i = Intent(...)
i.putExtra(key1, value1)
i.putExtra(key2, value2)
startActivity( i )
```

- Launching new Activities
  - When no response needed:
    `startActivity(intent)`

On Tap For Today

- Starting A Second Activity

- Passing Data To Activities
- Returning Data From Activities

- Practice

Start the Activity

- Launching new Activities
  - When no response needed:
    `startActivity(intent)`
  - When response is needed:
    `startActivityForResult(intent)`

Start the Activity

- Launching new Activities
  - When no response needed:
    `startActivity(intent)`
  - When response is needed:
    `startActivityForResult(intent)`
    - Response received in **`onActivityResult()`**

Start the Activity

- Launching new Activities
  - When no response needed:
    `startActivity(intent)`
  - When response is needed:
    ~~**`startActivityForResult(intent)`**~~
    - Response received in ~~**`onActivityResult()`**~~
- **`CallingActivity`** may have been killed when **`TargetActivity`** was running!

Passing Data Back

- The Target Activity creates a new `Intent`, puts data into it as an `extra`, and calls `setResult()` with the result status and intent as arguments

```
private fun setAnswerShownResult(isAnswerShown: Boolean) {
    val data = Intent()
    data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown)
    setResult(RESULT_OK, data)
}
```

Activity Launchers

- Launching new Activities
  - When no response needed:
    `startActivity(intent)`
  - When response is needed:
    ~~**`startActivityForResult(intent)`**~~
    - Response received in ~~**`onActivityResult()`**~~
- **`CallingActivity`** may have been killed when **`TargetActivity`** was running

Create ActivityResultContract

```
abstract class ActivityResultContract< I, O? >
    abstract fun createIntent(context: Context, input: I): Intent
    abstract fun parseResult(resultCode: Int, intent: Intent?): O?
```

Create ActivityResultCallback

```
abstract class ActivityResultContract< I, O? >
    abstract fun createIntent(context: Context, input: I): Intent
    abstract fun parseResult(resultCode: Int, intent: Intent?): O?

interface ActivityResultCallback< O? >
    abstract fun onActivityResult( result: O? )
```

Create ActivityResultLauncher

```
abstract class ActivityResultContract< I, O? >
    abstract fun createIntent(context: Context, input: I): Intent
    abstract fun parseResult(resultCode: Int, intent: Intent?): O?

interface ActivityResultCallback< O? >
    abstract fun onActivityResult( result: O? )

abstract class ActivityResultLauncher< I >
    abstract fun launch( input: I )
```

ActivityResult Setup

```
val contract = object : ActivityResultContract< I, O? > {
    override fun createIntent(context: Context, input: I): Intent { ... }
    override fun parseResult(resultCode: Int, intent: Intent?): O? { ... }

val callback = object : ActivityResultCallback< O? > {
    override fun onActivityResult( result: O? ) { ... }

override fun onCreate() {
val launcher: ActivityResultLauncher< I > = registerForActivityResult(contract, callback)
}
```

Android Design Patterns

- Behavioral Patterns
  1. Command – UI Event Handling, Retrofit Request Callback, Activity Result Callback
  2. Observer – State, Flow, LiveData
  3. Template Method - IScreenSpec
- Creational Patterns
  4. Builder – Compose NavGraph, WorkRequest, Constraints, Retrofit
  5. Factory – ViewModelFactory
  6. Singleton – ViewModelProvider, Repository, Room Database
- Structural Patterns
  7. Decorator – View Model
  8. Façade – DAO, Repository

ActivityResult Use

```
val contract = object : ActivityResultContract< I, O? > {
    override fun createIntent(context: Context, input: I): Intent { ... }
    override fun parseResult(resultCode: Int, intent: Intent?): O? { ... }

val callback = object : ActivityResultCallback< O? > {
    override fun onActivityResult( result: O? ) { ... }

override fun onCreate() {
val launcher: ActivityResultLauncher< I > = registerForActivityResult(contract, callback)
}

// in some click listener
launcher.launch( input )
```

ActivityResult Explicit Example

```
val contract = object : ActivityResultContract< String, SamodelkinCharacter? > {
  override fun createIntent(context: Context, input: String): Intent =
    SamodelkinCharacterPickerActivity.newIntent(context, input)
  override fun parseResult(resultCode: Int, result: Intent?): SamodelkinCharacter? {
    if(resultCode != Activity.RESULT_OK) return null
    return SamodelinCharacterPickerActivity.unpackCharacter(result)
  }


val callback = object : ActivityResultCallback< SamodelkinCharacter? > {
  override fun onActivityResult( newCharacter: SamodelkinCharacter? ) {
    if(newCharacter != null) {
      displayedCharacter = newCharacter
      // trigger view update
    }
  }


val launcher: ActivityResultLauncher< String > =
        registerForActivityResult(contract, callback)

...

launcher.launch( "Thorn Drumheller" )
```

ActivityResult Implicit Example

```
val contract = object : ActivityResultContract< Int, Uri? > {
  override fun createIntent(context: Context, ringtoneType: Int): Intent =
    Intent(RingtoneManager.ACTION_RINGTONE_PICKER).apply {
      putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE, ringtoneType)
    }
  override fun parseResult(resultCode: Int, result: Intent?): Uri? {
    if(resultCode != Activity.RESULT_OK) return null
    return result?.getParceableExtra(RingtoneManager.EXTRA_RINGTONE_PICKED_URI)
  }


val callback = object : ActivityResultCallback< Uri? > {
  override fun onActivityResult( newRingtone: Uri? ) {
    if(newRingtone != null) {
      // change ringtone
    }
  }


val launcher: ActivityResultLauncher< String > =
        registerForActivityResult(contract, callback)
...
launcher.launch( RingtoneManager.TYPE_ALARM )
```

ActivityResult Flow

```
callingActivity.onCreate()
   -> launcher = registerForActivityResult(contract, callback)
   -> launcher.launch( input )
   -> contract.createIntent()
   -> onStop()

targetActivity.onCreate()
   -> setResult()
   -> finish()

callingActivity.onResume()
   -> contract.parseResult
   -> callback.onActivityResult
```

ActivityResult Flow

```
callingActivity.onCreate()
  -> launcher = registerForActivityResult(contract, callback)
  -> launcher.launch( input )
  -> contract.createIntent()
  -> onStop()

targetActivity.onCreate()
  -> setResult()
callingActivity.onDestroy()
  -> finish()

callingActivity.onCreate()
  -> registerForActivityResult(contract, callback)
  -> contract.parseResult
  -> callback.onActivityResult
```

On Tap For Today

- Starting A Second Activity
- Passing Data To Activities
- Returning Data From Activities
- Practice

To Do For Next Time

- Lab07 due Fri Mar 10
- Alpha Release due Mon Mar 13 – have NavGraph in place
- A2 due Tue Mar 14
- Lab08 due Fri Mar 17
- Alpha Feedback due Fri Mar 17
- !!! Spring Break !!!
- Lab09 due Tue Mar 28