31. Instant Run
32~33. Event Handling
34~35. Gesture Detection

클라우드플랫폼개발랩
조영일

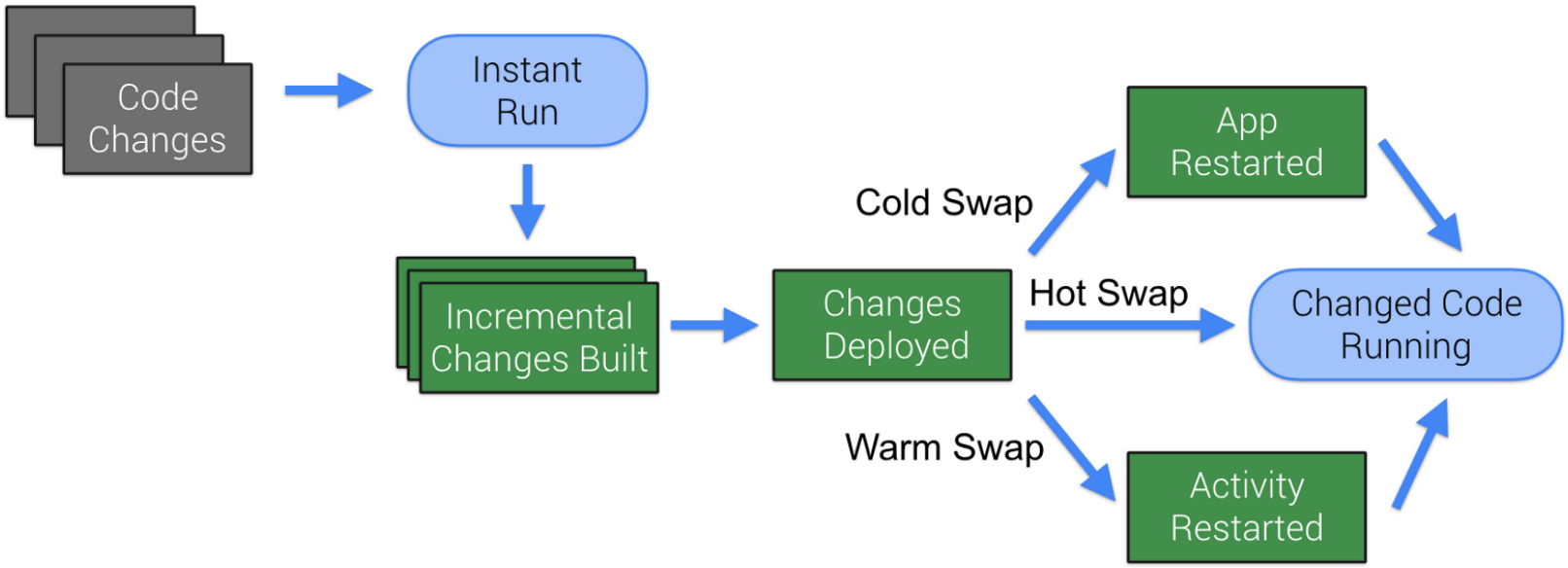
Instant Run

- 핫스왑이나 핫디플로이와 비슷한 개념
- 코드 컴파일 > Dex 포맷 변환, APK 패키징, 설치의 과정을 최대한 단축시키는 작업

Instant Run

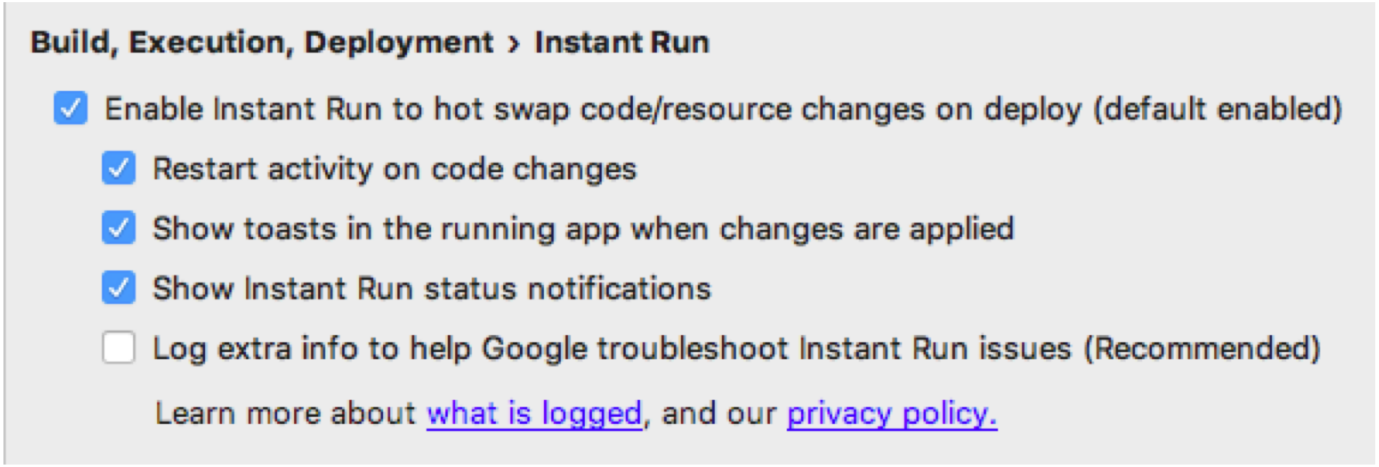
- Swapping Level
  - Hot Swapping
    - 기존 메소드 구현 내부의 코드가 변경될 때
    - 해당 메소드가 다음에 호출되면 신규 코드가 사용됨
    - Activity가 재시작하지 않음
  - Warm Swapping
    - 리소스가 변경될 때
    - 현재 실행 중인 activity가 재시작됨
  - Cold Swapping
    - 코드에 구조적인 변화가 발생할 때
    - 가끔은 이미지 리소스가 추가될 때
    - 앱이 재시작됨

Instant Run


Code Changes
Instant Run
Incremental Changes Built
Changes Deployed
Cold Swap
App Restarted
Hot Swap
Changed Code Running
Warm Swap
Activity Restarted

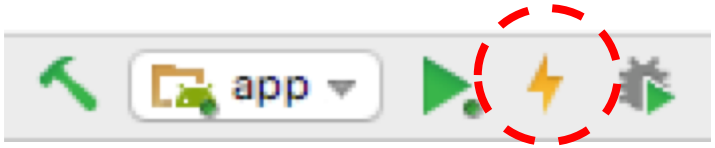
Instant Run

- 활성화
  - Preferences → Build, Execution, Deployment → Instant Run



Instant run

- 사용
  - Run 버튼과 Debug 버튼 사이에 번개 버튼
  - 한 번 실행한 후에 사용 가능해짐



Event Handling

- Input events
- Event queue
- Event notification with information
- Event listener, callback

Event Handling

- 이벤트 리스너의 종류
  - onClickListener
  - onLongClickListener
  - onTouchListener
  - onCreateContextMenuListener
  - onFocusChangeListener
  - onKeyListener

Event Handling

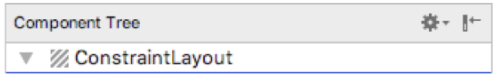
- 리소스에 지정
  - <Button
    - Android:onClick=“buttonClick”/>
  - 간단하지만 다양한 옵션을 제공하지 못함

Event Handling

- 예제 프로젝트 생성
  - SDK API 14 선택
  - Empty Activity 선택
  - 나머지는 default나 자유롭게 선택

Event Handling

- app → res → layout → activity_main.xml
  - Design > Component Tree에서 ContraintLayout 확인



  - toolbar에서 AutoConnect 확인

  - Button 추가
    - myButton / “Press Me”
    - 표지판 버튼⚠을 클릭, Fix 버튼을 클릭하여 리소스로 등록
  - TextView의 ID를 statusText로 변경

Event Handling

- listener registration
  - myButton
  - java > 패키지명 > MainActivity

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    myButton.setOnClickListener(object: View.OnClickListener {
        override fun onClick(v: View?) {
            statusText.text = "Button clicked"
        }
    })
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    myButton.setOnClickListener { it: View!
        statusText.text = "Button clicked"
    }
```

Event Handling

- event consumption / interception
  - onLongClickEvent의 경우, 리턴값을 지정해야 함
  - Android runtime에게 이벤트를 소비했는지의 여부를 알려줄 필요가 있음
  - false가 반환되면 다른 이벤트 리스너에게 전달
- 예제

```
myButton.setOnLongClickListener { it: View!
    statusText.text = "Long button clicked"
    ^setOnLongClickListener true
}
```

  - false로 바꿔서 테스트

Multi-touch Event Handling

- 사용법
  - myLayout.setOn**Touch**Listener { **v**: View, **m**: MotionEvent **->**
    - …
    - true
  - }
- MotionEvent
  - 이벤트 정보를 포함하는 객체
  - 터치의 위치, 수행된 액션의 타입
  - 멀티 터치 정보

Mutil-touch Event Handling

- MotionEvent
  - getActionMasked()
  - ACTION_DOWN → ACTION_MOVE → ACTION_UP
- 멀티 터치
  - getActionIndex()
  - ACTION_POINTER_DOWN → ACTION_POINTER_UP
- 사용법
  - myLayout.setOnTouchListener { v: View, m: MotionEvent ->
    - val pointerCount = m.pointerCount
    - val pointerId = m.getPointerId(0)
    - true
  - }

Multi-touch Event Handling

- 예제 프로젝트 생성
  - SDK API 14 선택
  - Empty Activity 선택
  - 나머지는 default나 자유롭게 선택
  - app → res → layout → activity_main.xml
    - 기존 textView 제거하고
    - textView 2개 배치
      - 각각 위쪽에서 16dp, 32dp 마진
      - ID는 textView1, textView2로 변경
      - 텍스트는 Touch One Status와 Touch Two Status로 변경
        - 경고 아이콘을 클릭하여 리소스로 뽑을 것
    - ConstraintLayout의 ID를 activity_motion_event로 변경

Multi-touch Event Handling

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    activity_motion_event.setOnTouchListener { v: View, m: MotionEvent ->
        handleTouch(m)
        ^setOnTouchListener true
    }
}
```

```kotlin
private fun handleTouch(m: MotionEvent) {
    val pointerCount = m.pointerCount
    for (i in 0 until pointerCount) {
        val x = m.getX(i)
        val y = m.getY(i)
        val id = m.getPointerId(i)
        val action = m.actionMasked
        val actionIndex = m.actionIndex
        val actionString: String

        when (action) {
            MotionEvent.ACTION_DOWN -> actionString = "DOWN"
            MotionEvent.ACTION_UP -> actionString = "UP"
            MotionEvent.ACTION_POINTER_DOWN -> actionString = "POINTER_DOWN"
            MotionEvent.ACTION_POINTER_UP -> actionString = "POINTER_UP"
            MotionEvent.ACTION_MOVE -> actionString = "MOVE"
            else -> actionString = ""
        }

        val touchStatus = "Action: $actionString Index: $actionIndex ID: $id X: $x, Y: $y"
        if (id == 0) {
            textView1.text = touchStatus
        } else {
            textView2.text = touchStatus
        }
    }
}
```

Gesture Detection

- Double Tap, Swipe(Fling) 등의 일반적인 제스쳐
- 과정이 좀 복잡한 편
  1. GestureDetector.OnGestureListener와 GestureDetector.OnDoubleTapListener 인터페이스를 Activity에서 구현
  2. GestureDetectorCompat 인스턴스를 생성
  3. GestureDetectorCompat.setOnDoubleTapListener() 호출
     - 선택적
  4. onTouchEvent 콜백 구현

Gesture Detection

- 예제
  - 기본 설정으로 프로젝트 생성
  - Layout에서 Hello World textView의 ID를 gestureStatusText로 지정

Gesture Detection

- GestureDetector 관련 인터페이스를 Activity에서 구현
  - MainActivity에 GestureDetector 관련 인터페이스 추가

```
class MainActivity : AppCompatActivity(),
        GestureDetector.OnGestureListener, GestureDetector.OnDoubleTapListener {
```

  - Code > Implement Methods 메뉴를 통해 몇 가지 메소드 구현

```
override fun onShowPress(e: MotionEvent?) {
    gestureStatusText.text = "onShowPress: " + e
}

override fun onSingleTapUp(e: MotionEvent?): Boolean {
    gestureStatusText.text = "onSingleTapUp: " + e
    return true
}

override fun onDown(e: MotionEvent?): Boolean {
    gestureStatusText.text = "onDown: " + e
    return true
}

override fun onFling(e1: MotionEvent?, e2: MotionEvent?,
                     velocityX: Float, velocityY: Float): Boolean {
    gestureStatusText.text = "onFling: " + e1 + " " + e2 + " " +
            velocityX + " " + velocityY
    return true
}

override fun onScroll(e1: MotionEvent?, e2: MotionEvent?,
                      distanceX: Float, distanceY: Float): Boolean {
    gestureStatusText.text = "onScroll: " + e1 + " " + e2 + " " +
            distanceX + " " + distanceX
    return true
}
```

```
override fun onLongPress(e: MotionEvent?) {
    gestureStatusText.text = "onLongPress: " + e
}

override fun onDoubleTap(e: MotionEvent?): Boolean {
    gestureStatusText.text = "onDoubleTap: " + e
    return true
}

override fun onDoubleTapEvent(e: MotionEvent?): Boolean {
    gestureStatusText.text = "onDoubleTapEvent: " + e
    return true
}

override fun onSingleTapConfirmed(e: MotionEvent?): Boolean {
    gestureStatusText.text = "onSingleTapConfirmed: " + e
    return true
}
```

Gesture Detection

- GestureDetectorCompat 인스턴스를 생성
- GestureDetectorCompat.setOnDoubleTapListener() 호출

```
var gDetector: GestureDetectorCompat? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    this.gDetector = GestureDetectorCompat( context: this, listener: this)
    gDetector?.setOnDoubleTapListener(this)
}
```

Gesture Detection

- onTouchEvent 콜백 구현

```
override fun onTouchEvent(event: MotionEvent?): Boolean {
    this.gDetector?.onTouchEvent(event)
    return super.onTouchEvent(event)
}
```

Custom Gesture & Pinch Recognition

- Custom Gesture
  - Gesture Builder를 이용하여 gesture file을 앱에 패키징
  - 이 데이터는 이후 사용자 gesture와 일치하는 것을 탐색하기 위해 GestureLibrary에 적재됨
  - GestureLibrary는 가장 비슷한 gesture를 탐색하여 ArrayList로 반환

Custom Gesture & Pinch Recognition

- GestureBuilder 앱을 이용해 gestures 파일 생성
  - Android에서 공식 지원하지 않음 (since 2012)
  - https://android.googlesource.com/platform/development/+/master/apps/GestureBuilder/
- gestures 파일 복사
  - View → Tool Windows → Device File Explorer 이용
  - SD카드에 저장된 gestures 파일을 PC에 저장
  - Studio의 res/raw/gestures로 복사
    - File → New → Directory로 raw 디렉토리 생성
    - gestures 파일은 Finder에서 직접 복사

Custom Gesture & Pinch Recognition

- layout.xml

```
<android.gesture.GestureOverlayView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/gOverlay"
    android:layout_gravity="center_horizontal">
</android.gesture.GestureOverlayView>
```

```
class MainActivity : AppCompatActivity(), GestureOverlayView.OnGesturePerformedListener {
    var gLibrary: GestureLibrary? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        gestureSetup();
    }

    private fun gestureSetup() {
        gLibrary = GestureLibraries.fromRawResource( context: this, R.raw.gestures)
        if (gLibrary?.load() == false) {
            finish()
        }
        gOverlay.addOnGesturePerformedListener( listener: this)
    }

    override fun onGesturePerformed(overlay: GestureOverlayView?, gesture: Gesture?) {
        val predictions = gLibrary?.recognize(gesture)
        predictions?.let { it: ArrayList<Prediction!>
            if (it.size > 0 && it[0].score > 1.0) {
                val action = it[0].name
                Toast.makeText( context: this, action, Toast.LENGTH_SHORT).show()
            }
        }
    }
```

Custom Gesture & Pinch Recognition

- layout.xml

```
<android.gesture.GestureOverlayView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/gOverlay"
    android:gestureColor="#00000000"
    android:uncertainGestureColor="#00000000"
    android:layout_gravity="center_horizontal">
</android.gesture.GestureOverlayView>
```

Custom Gesture & Pinch Recognition

- Pinch Gesture
  - ScaleGestureDetector 이용

- 예제
  - Default textView의 ID를 myTextView로 변경

```
class MainActivity : AppCompatActivity() {
    var scaleGestureDetector: ScaleGestureDetector? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        scaleGestureDetector = ScaleGestureDetector( context: this, MyOnScaleGestureListener())
    }
    override fun onTouchEvent(event: MotionEvent): Boolean {
        scaleGestureDetector?.onTouchEvent(event)
        return true
    }
    inner class MyOnScaleGestureListener: ScaleGestureDetector.SimpleOnScaleGestureListener() {
        override fun onScale(detector: ScaleGestureDetector): Boolean {
            val scaleFactor = detector.scaleFactor
            if (scaleFactor > 1) {
                myTextView.text = "Zooming out"
            } else {
                myTextView.text = "Zooming in"
            }
            return true
        }
        override fun onScaleBegin(detector: ScaleGestureDetector): Boolean {
            return true
        }
        override fun onScaleEnd(detector: ScaleGestureDetector?) {
        }
    }
```


THANK YOU FOR LISTENING
ANY QUESTIONS??
makeameme.org