03

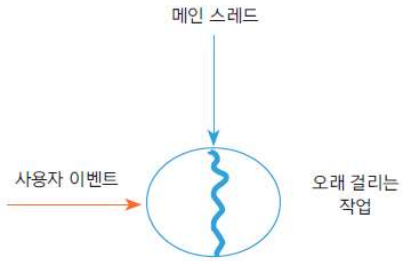
03-3. ANR

액티비티

13-3 액티비티 ANR 문제와 코루틴

ANR 문제란?

- ANR은 액티비티가 응답하지 않는 오류 상황
- 시스템에서 액티비티를 실행하는 수행 흐름을 메인 스레드 또는 화면을 출력하는 수행 흐름이라는 의미에서 UI 스레드라고 합니다.



13-3 액티비티 ANR 문제와 코루틴

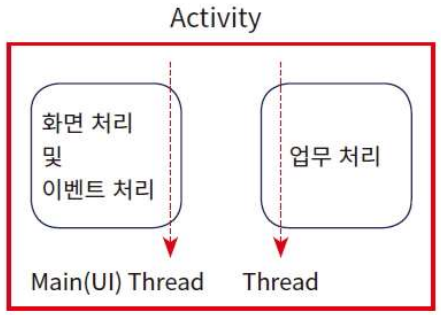
- ANR 문제를 해결하는 방법은 액티비티를 실행한 메인 스레드 이외에 실행 흐름(개발자 스레드)을 따로 만들어서 시간이 오래 걸리는 작업을 담당하게 하면 됩니다.
- 이 방법으로 대처하면 ANR 오류는 해결되지만 화면을 변경할 수 없다는 또 다른 문제


```
android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that
created a view hierarchy can touch its views.
```

Thread

Thread를 이용한 ANR 문제 해결방법

- 스레드를 만드는 방법은 Thread 클래스를 상속받아 작성하는 방법이 있고 Runnable 인터페이스를 구 현하여 작성하는 방법이 있다.



Thread

- Thread 상속

```
class MyThread : Thread() {
    override fun run() {

    }
}
```

```
val thread = MyThread()
thread.start()
```

- Runnable 구현

```
class MyThread : Runnable {
    override fun run() {

    }
}
```

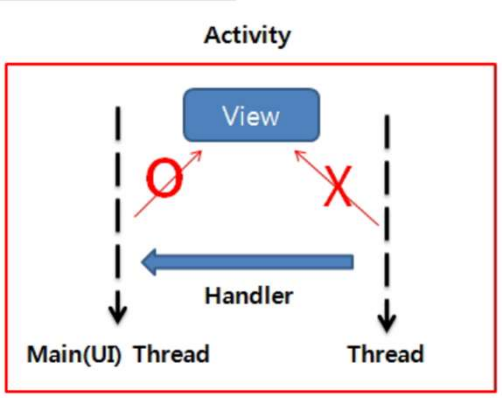
```
val thread = Thread(MyThread())
thread.start()
```

Handler

- 스레드-핸들러 구조
- UI 스레드가 아닌, 개발자 스레드에서 런타임 때 액티비티의 화면 출력 요소인 뷰에 접근하면 에러가 발생

```
inner class MyThread : Thread() {
    override fun run() {
        var sum = 0
        for (i in 1..10) {
            sum += i
            textView.setText("sum:$sum")//error……
            try {
                Thread.sleep(1000)
            } catch (e: InterruptedException) {
            }

        }
    }
}
```



Handler

post ( ) 함수를 이용

```
var handler = Handler()
```

- 뷰와 관련된 작업을 담당하 는 Runnable을 구현한 클래스를 준비

```
inner class UIUpdate : Runnable {
    override fun run() {
        textView.text = "sum:$sum"
    }
}
```

- Handler의 post ( ) 함수를 호출해 UI 스레드에게 작업을 의뢰

```
inner class MyThread : Thread() {
    override fun run() {
        var sum = 0
        for (i in 1..10) {
            sum += i
            handler.post(UIUpdate())
            try {
                Thread.sleep(1000)
            } catch (e: InterruptedException) {
            }

        }
    }
}
```

Handler

sendMessage ( ) 함수를 이용

- sendMessage(Message msg): UI 스레드에 의뢰
- sendMessageAtFrontOfQueue(Message msg): 이번 의뢰를 가장 먼저 처리
- sendMessageAtTime(Message msg, long uptimeMillis): 지정된 시간에 수행해 달라는 요청
- sendMessageDelayed(Message msg, long delayMillis): 의뢰를 바로 처리하지 말고 지정 시간 후에 수행해 달라는 요청
- sendEmptyMessage(int what): 데이터 전달 없이 의뢰하는 경우

Message 객체는 뷰작 업을 의뢰할 때 UI 스레드에 넘기는 데이터를 담는 객체

- what: int 형 변수로 구분자. 개발자가 임의의 숫자로 요청을 구분하기 위해 사용
- obj: UI 스레드에 넘길 데이터. Object 타입의 변수
- arg1, arg2: UI 스레드에 넘길 데이터, int 형

Handler

```
inner class MyThread : Thread() {
    override fun run() {
        try {
            var count = 10
            while (loopFlag) {
                Thread.sleep(1000)
                if (isRun) {
                    count--
                    var message = Message()
                    message.what = 1
                    message.arg1 = count
                    handler.sendMessage(message)
                    if (count == 0) {
                        message = Message()
                        message.what = 2
                        message.obj = "Finish!!"
                        handler.sendMessage(message)
                        loopFlag = false
                    }
                }
            }
        } catch (e: Exception) {

        }
    }
}
```

Handler

- sendMessage( ) 함수로 요청을 처리하려면 Handler 클래스의 서브 클래스를 정의

```
var handler: Handler = object : Handler() {
    override fun handleMessage(msg: Message) {
        if (msg.what == 1) {
            textView.text = msg.arg1.toString()
        } else if (msg.what == 2) {
            textView.text = msg.obj as String
        }
    }
}
```

AsyncTask

- 스레드-핸들러의 추상화 개념
- AyncTask를 상속받는 클래스를 작성
- API 30에서 deprecated

- doInBackground(Params... params): 스레드에 의해 처리될 내용
- onPreExecute(): AsyncTask의 작업을 시작하기 전에 호출.
- onPostExecute(Result result): AsyncTask의 모든 작업이 완료된 후 가장 마지막에 한 번 호출. doInBackground() 함 수의 최종 값을 받기 위해 사용
- onProgressUpdate(Progress... values): doInBackground() 함수에 의해 처리되는 중간중간 값을 받아 처리하기 위해 호출. doInBackground() 함수에서 publishProgress() 함수로 넘긴 값이 전달

```
class MyAsyncTask : AsyncTask<Void, Int, String>() {
    override fun doInBackground(vararg params: Void): String {
        //...
    }

    override fun onProgressUpdate(vararg values: Int?) {
        super.onProgressUpdate(*values)
    }

    override fun onPostExecute(values: String) {

    }
}
```

AsyncTask

```
class MyAsyncTask : AsyncTask<Void, Int, String>()
```

- 첫 번째 타입: 백그라운드 작업을 위한 doInBackground() 함수의 매개변수 타입
- 두 번째 타입: doInBackground() 함수 수행에 의해 발생한 데이터를 publishProgress() 함수를 이용해 전달하는데 이때 전달 데이터 타입.
- 세 번째 타입: onPostExecute() 함수의 매개변수 타입과 동일하게 지정 doInBackground() 함수의 반환형

- AsyncTask 클래스를 수행하려면 클래스를 생성한 다음 execute( ) 함수를 호출

```
val task = MyAsyncTask()
task.execute()
```

13-3 액티비티 ANR 문제와 코루틴

코루틴으로 ANR 오류 해결

- 코루틴이란?
- 비동기 경량 스레드
- 프로그래밍 언어에서 제공하는 기능
- 코루틴을 사용하라고 권하면서 다음과 같은 장점이 있다고 소개
  - 경량입니다.
  - 메모리 누수가 적습니다.
  - 취소 등 다양한 기능을 지원합니다.
  - 많은 제트팩 라이브러리에 적용되어 있습니다.

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare