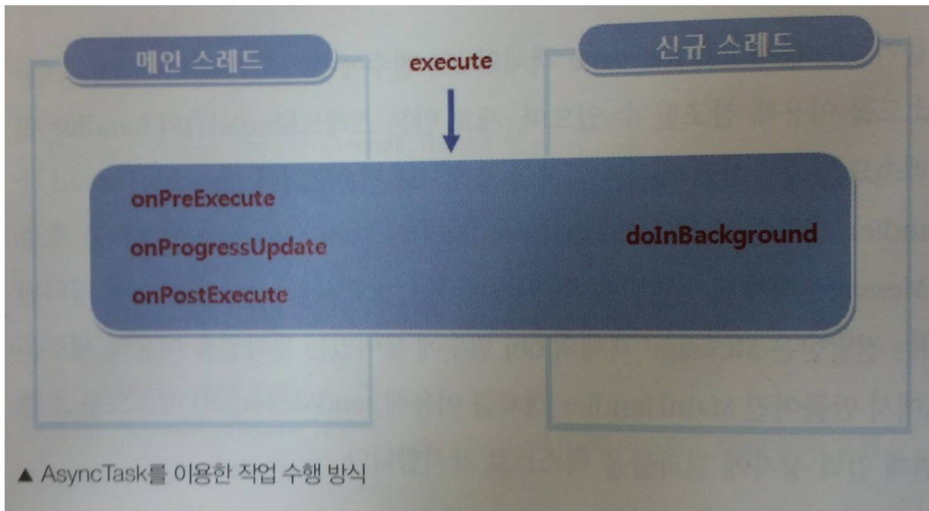


목차

1. AsyncTask.....	2
2. AsyncTask 동작 이미지	3
3. AsyncTask 구현 예제	4
3.1 실행 방법.....	4
4. AsyncTask 방식.....	5
5. Reference.....	7

1. AsyncTask

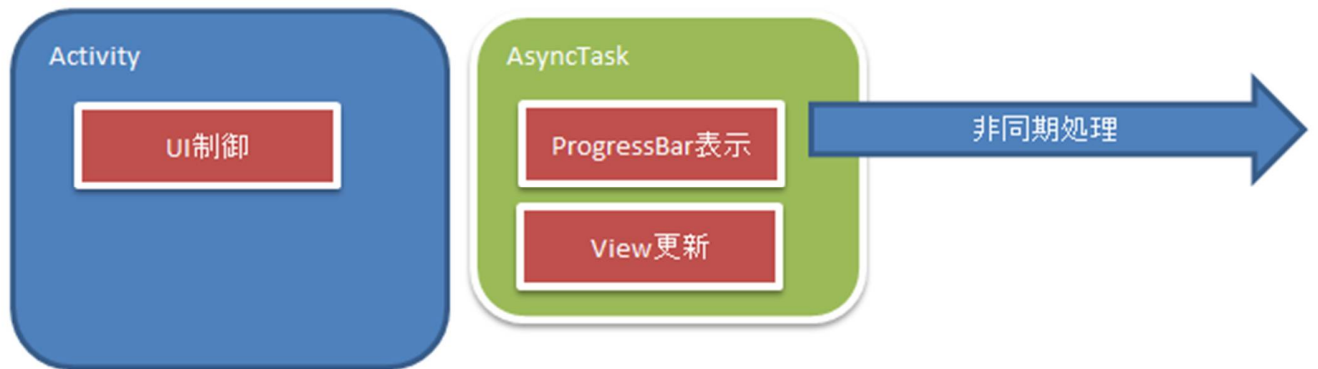


안드로이드 AsyncTask 사용법에 대해 알아 보겠습니다. 안드로이드에는 UI 를 총괄하는 메인 Thread 와 일반 Thread 가 존재 하는데요. 메인 Thread 외에는 일반 Thread 들이 안드로이드 UI 화면을 처리할 수 없습니다. 그렇기 때문에 메인 Thread 와 일반 Thread 를 잘 핸들링 해서 사용해야 하는데, 여간 번거로운 일이 아닙니다.

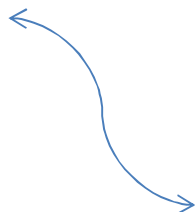
그렇기 때문에, Android 에서는 AsyncTask 라는 객체를 지원하는데요. AsyncTask 는 UI 처리 및 Background 작업을 하나의 클래스에서 작업 할 수 있게 지원해 줍니다. 쉽게말해 메인 Thread 와 일반 Thread 를 가지고 Handler 를 사용하여 핸들링하지 않아도 AsyncTask 객체하나로 편하게 UI 를 수정 할 수 있고, Background 작업을 진행 할 수 있습니다. 각각의 주기마다 CallBack 메서드를 사용해서 처리하는 장점이 있습니다.

- AsyncTask 는 Handler 와 Thread 를 이용하지 않고 UI Thread 를 이용하여 Background 작업을 수행.
- AsyncTask 는 UI Thread 에서 실행되기 때문에 UI 의 변경도 가능하다.

2. AsyncTask 동작 이미지

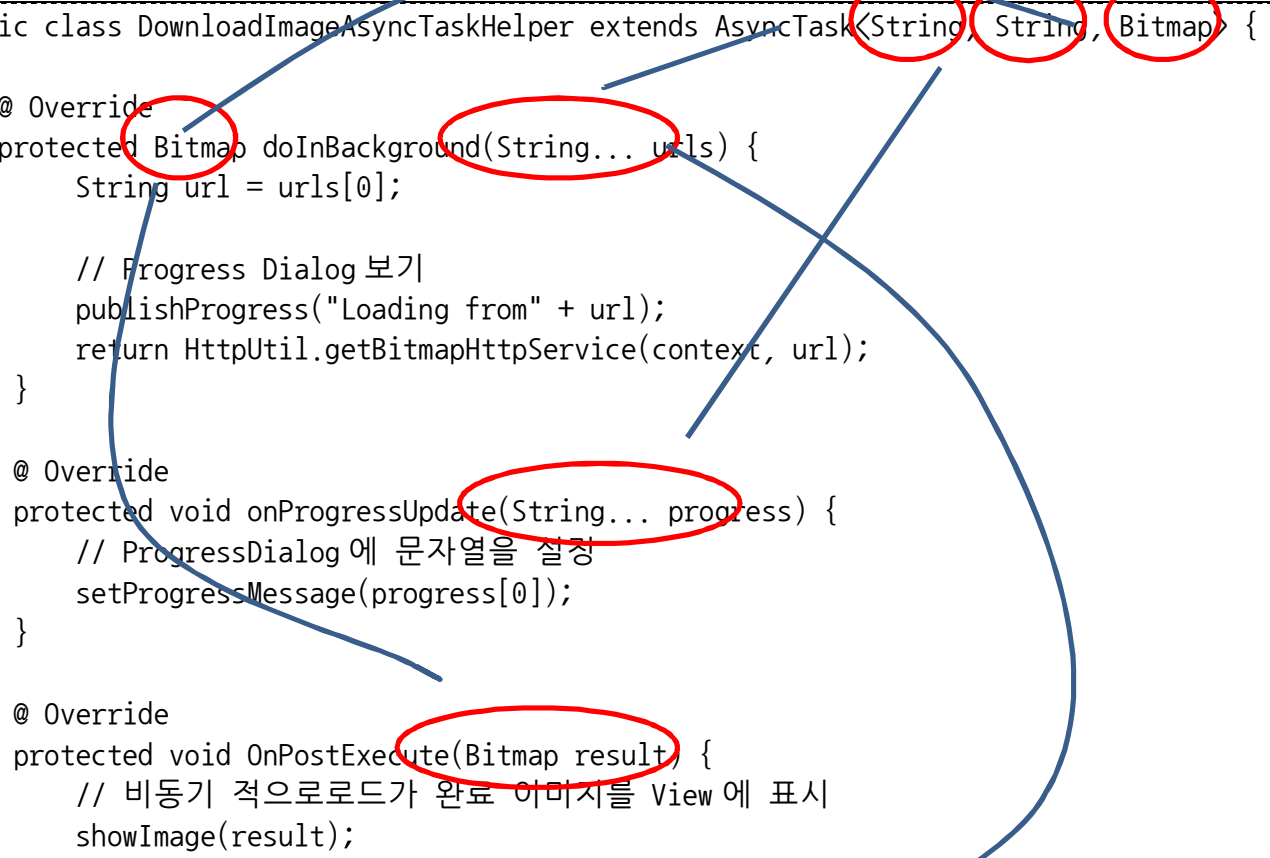


같은 화면의 UI를 제어하는 코드가 여러 클래스에 포함되어 있습니다.



3. AsyncTask 구현 예제

```
public class DownloadImageAsyncTaskHelper extends AsyncTask<String, String, Bitmap> {  
  
    @ Override  
    protected Bitmap doInBackground(String... urls) {  
        String url = urls[0];  
  
        // Progress Dialog 보기  
        publishProgress("Loading from" + url);  
        return HttpUtil.getBitmapHttpService(context, url);  
    }  
  
    @ Override  
    protected void onProgressUpdate(String... progress) {  
        // ProgressDialog 에 문자열을 설정  
        setProgressMessage(progress[0]);  
    }  
  
    @ Override  
    protected void onPostExecute(Bitmap result) {  
        // 비동기 적으로로드가 완료 이미지를 View 에 표시  
        showImage(result);  
    }  
}
```



3.1 실행 방법

```
DownloadImageAsyncTaskHelper task = new DownloadImageAsyncTaskHelper();  
task.execute("http://www.naver.com");
```

4. AsyncTask 방식

```
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;

public class TestAsyncTaskActivity extends Activity {

    private MyAsyncTask myAsyncTask;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        myAsyncTask = new MyAsyncTask();
        myAsyncTask.execute("80", "90", "100", "110");
    }

    // AsyncTask 클래스는 항상 Subclassing 해서 사용 해야 함.
    // 사용 자료형은
    // background 작업에 사용할 data 의 자료형: String 형
    // background 작업 진행 표시를 위해 사용할 인자: Integer 형
    // background 작업의 결과를 표현할 자료형: 인자를 사용하지 않은 경우 Void 로 지정.
    public class MyAsyncTask extends AsyncTask<String, Void, String> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected String doInBackground(String... ) {

            String sum = "";

            if(params != null){
                for(String s : params){
                    sum += s;
                }
            }

            return sum;
        }
    }
}
```

```

@Override
protected void onPostExecute(String result) {
    super.onPostExecute(result);

    if(result != null){
        Log.d("ASYNC", "result = " + result);
    }

}

@Override
protected void onCancelled() {
    super.onCancelled();
}

}
}

```

1. `onPreExecute()` : Background 작업 시작전에 UI 작업을 진행 한다.

```

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

```

2. `doInBackground()` : Background 작업을 진행 한다.

```

@Override
protected String doInBackground(String... params) {
    super.onPreExecute();
}

```

3. `onPostExecute()` : Background 작업이 끝난 후 UI 작업을 진행 한다.

```

@Override
protected void onPostExecute(String result) {
    super.onPreExecute();
}

```

FLOW 를 살펴 보자면,

[`onPreExecute()`] -> [`doInBackground()`] -> [`onPostExecute()`] 순서가 됩니다.

5. Reference

<http://itzone.tistory.com/464>

<http://tigerwoods.tistory.com/28>