Android
028 Loading Library

1. Loading Library for Android

1. 안드로이드에서는 처리시간이 긴 Task를 처리하기 위한 다양한 로딩 라이브러리들을 사용할 수 있다

1) load images

a. AUIL(Universal Image Loader)
b. Glide
c. picasso

2) load something on the network

a. retrofit
b. volley
c. aQuery

(okHttp - 로딩라이브러리는 아니다. 비동기 처리를 따로해줘야한다.)

2. Serial excution for AsyncTasks

1. AsyncTask 처리시 3.2 이상버전부터는 두개이상의 AsyncTask 실행시 순서대로 실행되는 Serial excution 이 일어나기 때문에 리스트와 같이 동시에 여러개의 로딩이 일어나는 경우는 적합하지 않다

- 버전별 AsyncTask 처리

Options	1.5	1.6	2.3	3.2
CORE_POOL_SIZE	1	5	5	AsyncTask.SerialExecutor에 의해 직렬 처리
MAXIMUM_POOL_SIZE	10	128	128	
KEEP_ALIVE	10	10	1	

- 병렬처리를 위한 코드추가

```
if (Build.VERSION.SDK_INT>=Build.VERSION_CODES.HONEYCOMB) {
    asynctask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
} else {
    asyncask.execute();
}
```


3.1. Image Loader Comparison

1. 아래는 많이 사용되는 이미지 로더간의 성능비교표이다

Library 종류	이미지 용량(KB)	최초		디스크 캐시		메모리 캐시		스크롤 반복
		메모리(MB)	속도(ms)	메모리(MB)	속도(ms)	메모리(MB)	속도(ms)	최대 할당 메모리 (MB)
Glide	301	4.6	2603	2.2	159	0.1	1	115
	563	2.9	526	1.4	45	0.1	1	
	219	1.8	575	1.1	36	0	2	
	456	0.9	715	1.3	54	0.1	1	
	238	2.3	407	1.2	37	0.1	1	
	556	1.9	507	1.3	48	0.1	1	
Picasso	301	9.3	706	9.5	542	0.2	0	65
	563	9.4	1021	9.4	355	0.1	1	
	219	9.5	496	9.5	333	0.1	0	
	456	9.4	566	9.4	344	0.1	1	
	238	9.6	1117	9.4	339	0.1	0	
	556	9.5	970	9.4	361	0.1	0	
AUIL (Android-Universal-Image-Loader)	301	7.6	1924	7.4	169	7.4	113	65
	563	7.2	2361	7.5	142	0	130	
	219	7.3	1084	7.5	99	7.5	134	
	456	7.4	2089	7.5	140	7.6	159	
	238	7.3	1030	7.4	136	7.4	122	
	556	7.4	1968	7.5	149	7.5	1	
Volley	301	2.4	964	2.2	267	0.1	188	100
	563	3	795	2.4	227	0.1	382	
	219	2.4	481	2.2	178	0	1	
	456	2.8	1346	2.3	333	0.1	1	
	238	2.4	1436	2.1	333	0.1	0	
	556	3	1817	2.5	347	0.1	1	

기준:2015-02-09, 출처:http://dev2.prompt.co.kr/31

3.2. Image Loader Comparison

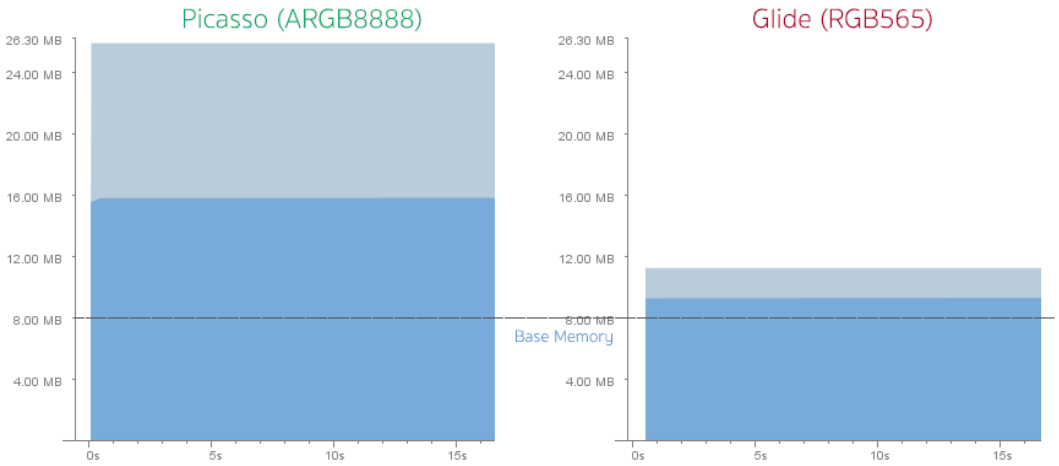
1. 비고

Library 종류	비고
Glide	빠른 속도와 안정성을 보장하고 이미지캐시용량이 작아서 스크롤 시에도 이미지가 빠르게 뜸
Picasso	옵션 설정에 따라 Glide와 동일한 성능 또는 경우에 따라 더 높은 성능을 낼 수 도 있다
AUIL (Universal-Image-Loader)	메모리 캐시 히팅률이 낮음, 스크롤 시 이전 이미지가 보여지는 현상 발생.
Volley	가끔 OOM으로 죽는 현상 발생, 빠르게 상하 스크롤시 이미지 거의 안 뜸.

출처:http://dev2.prompt.co.kr/31

2. Glide vs Picasso 메모리사용 (default 옵션시)
   - 사용하는 bitmap 포맷이 다르기때문에 메모리 사용량이 상이하다

Picasso의 경우 **resize 옵션**을 Glide의 default 사이즈와 같은 **768 x 432**로 하면 동일한 양의 메모리를 사용한다



Glide는 추가로
1. GIF
2. Thumbnail
3. Custom Animation
4. Custom load 를 지원한다

4. Network Loader Comparison

1. 이미지 로딩뿐만아니라 네트웍을 통한 데이터 전달시 사용된다
   1) retrofit
      - 설정이 단순해서 접근성이 뛰어나다
      - Restful에 요청을 위해 작성되는 인터페이스가 단순하며
        json 컨버터와의 연동으로 데이터를 객체화해서 사용할 수 있다

   2) volley
      - retrofit에 비해 설정이 복잡하다
      - 하지만 다양한 옵션을 사용할 수 있기때문에 익숙해지면 retrofit 보다
        뛰어난 성능을 발휘한다

   3) aQuery
      - 단순요청에 대한 응답속도는 가장 빠르다

2. 동일한 문제점
   - 셋다 대용량 파일 다운로드에 대한 out-of-memory 이슈가 있다
     이 부분은 DownloadManger를 통해 해결하는것이 권장된다

5. Glide and Picasso

1. 둘 다 설정 및 사용법이 아주 흡사하다
 - 공식홈
  Glid : https://github.com/bumptech/glide
  Picasso : http://square.github.io/picasso/
 - 라이브러리 설정

Glide
compile 'com.github.bumptech.glide:glide:3.7.0' //2016-10-22 최신

Picasso
compile 'com.squareup.picasso:picasso:2.5.2' // 2016-10-22 최신

- 소스코드

Glide
Glide.with(context).load("이미지 URI").into(View객체);

Picasso
Picasso.with(context).load("이미지 URI").into(View객체);

