http://lomohome.com/316
by Geunwon,Mo (mokorean@gmail.com)

Android 의 MapView (Google API) 정리하기.

원래 하나은행 스마트폰 뱅킹의 위치기반(LBS) 지점찾기는 WebView 에서 Google Map API 를 통하여 구현이 되어있었다.
아이폰에서는 이게 잘 돌아가는데... 안드로이드에서는 기계마다 되는것도 있고, 안되는것도 있고..
영 껄쩍지근 했다. (사실 이번에 출시한 갤럭시 S 에서 안돌아가는 이유가 가장 컸지..)

그래서 내친김에 WebView 에서 구현하지말고 MapView 로 구현해버리기로 했다.
이틀정도 작업한거라 고쳐야할 부분도 많고 (특히 Runnable 로 구현한 길게 누르기는...) 버그도 좀 있지만
일단 돌아가니, 이제까지 한것을 까먹지 않으려고 블로그에 정리를 해 둔다.

MapView 추가하기.

AndroidMenifest.xml 을 수정한다.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

윗줄부터 INTERNET 은 구글지도API 가 인터넷연결을 통하여 데이터를 받아오기때문에 추가해주어야하고
ACCESS_***_LOCATION 은 현재위치를 프로바이더(네트웍,GPS)를 통하여 받아오기 위해 추가해준다.

그 다음, <application> 태그 안쪽에 수정되어야 할 항목이다. 먼저,

```
<!-- 안드로이드 맵뷰를 사용하려면 라이브러리를 추가한다. -->
<uses-library android:name="com.google.android.maps" />
```

라이브러리를 사용함을 선언해준다. 그리고 액티비티 선언을 하나 추가해준다.

```
<!-- 지점찾기 맵 -->
<activity android:name=".BranchMapActivity"
android:screenOrientation="portrait">
      <intent-filter>
            <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
</activity>
```

다음은 레이아웃을 그려줄 branchmap.xml 에서 쓰인 맵뷰 부분의 선언이다.

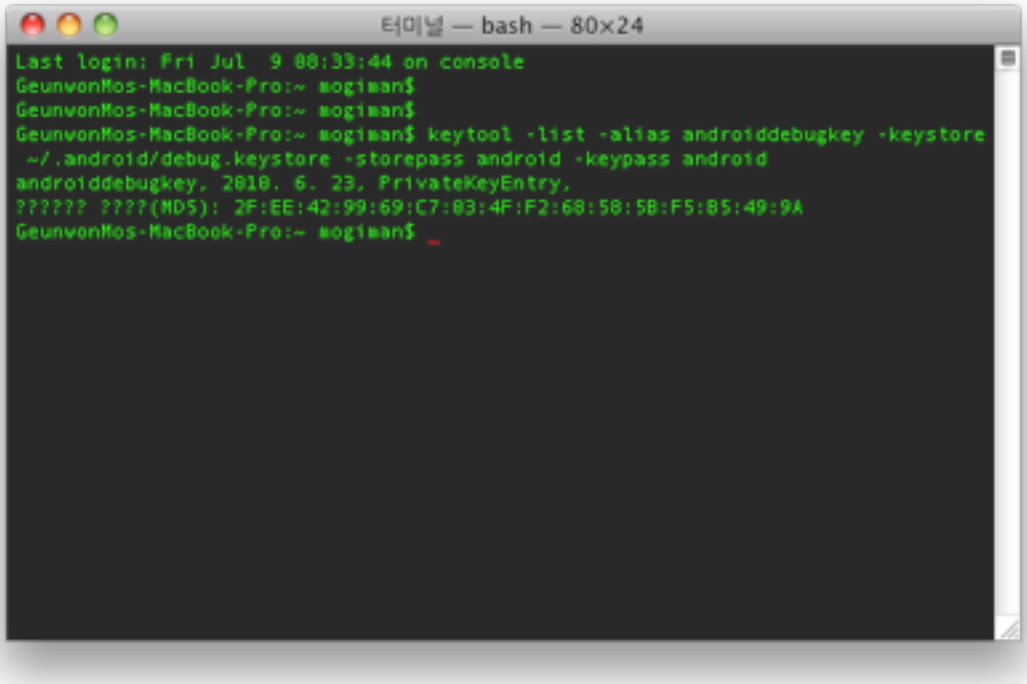
```
...
<com.google.android.maps.MapView
            android:id="@+id/mapView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:enabled="true"
            android:clickable="true"
            android:apiKey="0kiM******" /> <!-- API 키를 등록해야 동작한다. --
>
...
```

위에서 쓰인 android:apiKey 는 각 개발머신에 따라 따로 받아서 적어넣어야한다.
API Key 를 넣지않으면 동작은 하지만 지도데이터를 받아오지 않는다.
여기서 따로 설명은 하지 않고, 다음의 링크를 따라가면 MD5 값을 가지고 구글 API 키를 받아오는법이 잘 설명이 되어있다.

http://www.mobileplace.co.kr/1070

참고로 나는 맥을 사용해서 개발을 진행하였기때문에 다음의 명령어로 MD5키를 받아왔다.

keytool -list -alias androiddebugkey -keystore ~/.android/debug.keystore -storepass android -keypass android



받아온다음 Google Map API 사이트 (http://code.google.com/intl/ko-KR/android/maps-api-signup.html)에서 API를 받아와서 XML 에 넣어주면 된다.

이제 맵뷰를 표시하는 핵심 클래스인 BranchMapActivity.java 의 내용중 맵뷰에 관련한 부분을 정리해본다.

```
public class BranchMapActivity extends MapActivity {
```

맵을 표시하는 액티비티는 MapActivity 를 상속받아 구현한다.

다음은 전역변수로 사용되어진 변수 중, 지도의 표시에 관련한 변수들이다.

```
private MapView mapView; //맵뷰 객체
private List<Overlay> listOfOverlays; //맵에 표시된 오버레이(레이어)들을 가지고 있는 리스트
private String bestProvider; //현재 위치값을 가져오기위한 프로바이더. (network, gps)

private LocationManager locM; //위치 매니저
private LocationListener locL; //위치 리스너
```

```
private Location currentLocation; //현재 위치
private MapController mapController; //맵을 줌시키거나, 이동시키는데 사용될 컨트롤러

private LocationItemizedOverlay overlayHere; //현재위치 마커가 표시되어질 오버레이
private LocationItemizedOverlay overlayBranch; //지점위치 마커들이 표시되어질 오버레이
private List<BranchInfoDTO> brList; //지점리스트
```

다음은 onCreate 메소드에서 맵뷰에 관련한 부분이다.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);

	...

	setContentView(R.layout.branchmap); //맵액티비티 xml을 풀어헤친다.

	...

	overlayHere = null;
	overlayBranch = null; //각 오버레이 초기화

	...

	mapView = (MapView) findViewById(R.id.mapView); //맵뷰 객체를 가져온다.
	mapView.setBuiltInZoomControls(true); //줌인,줌아웃 컨트롤을 표시한다.

	mapController = mapView.getController(); //맵컨트롤러를 가져온다.
	mapController.setZoom(17); //초기 확대는 17정도로..

	//위치 매니저를 시스템으로부터 받아온다.
	locM = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

	//사용가능한 적절한 프로바이더를 받아온다.
	//network (보통 3G망,Wifi AP 위치정보)또는 gps 둘중 하나로 설정된다.
	bestProvider = locM.getBestProvider(new Criteria(), true);

	//기기에 가지고 있는 마지막 위치정보로 현재위치를 초기 설정한다.
	currentLocation = locM.getLastKnownLocation(bestProvider);

	//위치 리스너 초기화
	locL = new MyLocationListener();
	//위치 매니저에 위치 리스너를 셋팅한다.
	//위치 리스너에서 10000ms (10초) 마다 100미터 이상 이동이 발견되면 업데이트를 하려한다.
	locM.requestLocationUpdates(bestProvider, 10000, 100, locL);

	//처음에 한번 맵뷰에 그려준다.
	updateOverlay(currentLocation);
}
```

위에서 한번 언급된 MyLocationListener 는 액티비티 클래스안에 인너클래스로 구현한다.
리스너는 로케이션 매니저에 추가되어 GPS 나 네트워크로부터 위치정보 변경되는것을 감시하게 된다.

```
public class MyLocationListener implements LocationListener {

	@Override
	public void onLocationChanged(Location location) {
		//위치 이동이 발견되었을때 호출될 메소드.
		//위의 설정에서 10초마다 100미터 이상 이동이 발견되면 호출된다.
		updateOverlay(location);
```

```
	}

	@Override
	public void onProviderDisabled(String provider) {
		Log.d(LOG_TAG, "GPS disabled : " + provider);
	}

	@Override
	public void onProviderEnabled(String provider) {
		Log.d(LOG_TAG, "GPS Enabled : " + provider);
	}

	@Override
	public void onStatusChanged(String provider, int status, Bundle extras) {
		Log.d(LOG_TAG, "onStatusChanged : " + provider + " & status = "
				+ status);
	}

}
```

다음은 내가 구현한 지도그려주기 액티비티의 꽃이라 할수 있는 updateOveray 메소드이다.
요청을 받으면 Location 객체 (위치)를 기준으로 현재위치 마커를 찍고, 지점리스트를 HttpClient 를 통하여 통신해서 받아온후 지점들의 마커를 표시하게 된다.

```
protected void updateOverlay(Location location) {

	//기존에 화면에 찍어둔 오버레이 (마커들)을 싹 지운다.
	listOfOverlays = mapView.getOverlays(); //맵뷰에서 오버레이 리스트를 가져온다.
	if (listOfOverlays.size() > 0) {
		listOfOverlays.clear(); //오버레이가 있을때 싹 지워준다.
		Log.d(LOG_TAG, "clear overlays : " + listOfOverlays.size());
	} else {
		Log.d(LOG_TAG, "empty overlays");
	}

	//Location 객체를 가지고 GeoPoint 객체를 얻어내는 메소드
	GeoPoint geoPoint = getGeoPoint(location);
	//현재위치를 표시할 이미지
	Drawable marker;

	//실제 운영소스엔 분기하여 현재위치와 선택위치 이미지를 변경하게 되어있다.
	marker = getResources().getDrawable(R.drawable.icon_here);
	marker.setBounds(0, 0, marker.getIntrinsicWidth(),
marker.getIntrinsicHeight());

	//LocationItemizedOverlay 를 이용하여 현재위치 마커를 찍을 오버레이를 생성한다.
	overlayHere = new LocationItemizedOverlay(marker);
	//touch event 의 null pointer 버그를 방지하기 위해 마커를 찍고 바로 populate 시켜
준다.
	overlayHere.mPopulate();

	//현재위치를 GeoCoder 를 이용하여 대략주소와 위,경도를 Toast 를 통하여 보여준다.
	String geoString = showNowHere(location.getLatitude(),
location.getLongitude() , true);

	//현재위치 마커 정의
	OverlayItem overlayItem = new OverlayItem(geoPoint, "here", geoString);
	overlayHere.addOverlay(overlayItem); //현재위치 오버레이 리스트에 현재위치 마커를
넣는다.

	// 지점정보를 HTTP통신을 통해 서버에서 받아와서 전역변수인 brList (지점리스트)에 넣는다.
	// 성능을 고려하여 쓰레드로 구현이 되어있다.
```

```
		// 고다음 지점리스트 오버레이에 넣고 화면에 찍어주는 메소드.
		showBranchMarker(location.getLatitude(), location.getLongitude(),
				this.searchType, SEARCH_RANGE);


		// 맵뷰에서 터치이벤트를 받을 오버레이를 추가한다.
		// 특정지점을 오래 눌렀을때 특정 지점 기준으로 재검색을 하기 위하여 터치이벤트를 받아와야한
다.
		mapView.getOverlays().add(new MapTouchDetectorOverlay());

		// 마지막으로 생성된 오버레이레이어를 맵뷰에 추가한다.
		mapView.getOverlays().add(overlayHere);
		mapView.getController().animateTo(geoPoint); //현재위치로 화면을 이동한다.
		mapView.postInvalidate(); //맵뷰를 다시 그려준다.

}
```

조금 복잡하고 지저분하게 구성되어있어 퍼포먼스는 조금 떨어진다. 개선의 여지가 있다.
시간나면 수정해보자...

다음은 updateOverlay 메소드에서 사용되었던 getGeoPoint 메소드 전문이다.

```
private GeoPoint getGeoPoint(Location location) {
		if (location == null) {
			return null;
		}
		Double lat = location.getLatitude() * 1E6;
		Double lng = location.getLongitude() * 1E6;
		return new GeoPoint(lat.intValue(), lng.intValue());
}
```

별것 없다. 주의해야할점은 GeoPoint 객체는 위도, 경도 표시에 1E6 을 곱해줘야한다는것이다.

그리고 마커를 생성하고 오버레이에 표시, 그리고 마커를 눌렀을때 이벤트를 발생시키는 클래스이다.
인너클래스로 구현하였다.

```
protected class LocationItemizedOverlay extends
			ItemizedOverlay<OverlayItem> {
		private List<OverlayItem> overlays;

		public LocationItemizedOverlay(Drawable defaultMarker) { //오버레이 생성자
			//마커 이미지의 가운데 아랫부분이 마커에서 표시하는 포인트가 되게 한다.
			super(boundCenterBottom(defaultMarker));
			overlays = new ArrayList<OverlayItem>();
		}

		@Override
		protected OverlayItem createItem(int i) {
			return overlays.get(i);
		}

		@Override
		public int size() {
			return overlays.size();
		}

		public void addOverlay(OverlayItem overlay) {
			overlays.add(overlay);
			//null pointer 버그때문에 오버레이 아이템 추가후 가능한 빨리 populate 해줘야한
다.
			populate();
```

```
	}

	@Override
	protected boolean onTap(int index) {

		//마커를 눌렀을때 발생시킬 이벤트 메소드이다.

		if ("here".equals(overlays.get(index).getTitle())) {
			//현재 위치일 경우 간단한 토스트 메세지를 보여준다.
			Toast.makeText(getApplicationContext(),
					overlays.get(index).getSnippet(),
Toast.LENGTH_SHORT)
					.show();
		} else {
			//지점선택일 경우 다이얼로그를 통하여 지점정보를 보여준다.
			//'전화걸기' 버튼으로 지점으로 전화거는 기능도 추가되어있다.
			//맵뷰에 관련한 소스가 아니어서 이곳에서는 표시 하지 않는다.
			...
		}

		return true;
	}

	//외부에서 마커의 populate 를 해주기 위한 메소드.
	public void mPopulate() {
		populate();
	}
}
```

지점 정보를 HTTP 통신을 통해 가져오는 메소드이다.
HTTP 통신시 랙현상을 없애기위해 쓰레드로 구현을 해봤다.
근데 스레드가 생각한대로 동작하진 않는것 같다. 잘못쓰고 있는것일까... -_-

```
private void showBranchMarker(Double lat, Double lng, String searchType,
		String searchRange) {

	GetMapDataThread excuteThread = new GetMapDataThread(getMapdataHandler,
			lat, lng, searchType, searchRange);
	excuteThread.start();
}
```

실제 HTTP통신을 하는 클래스를 호출하는 쓰레드이다.
HTTP 통신 부분은 지도표시와 상관이 없기때문에 여기서 소스를 게시하지는 않는다.
다만 기존에 HTTPConnection 으로 구현되어있던 HTTP 통신을 HTTPClient 로 변경하니까
퍼포먼스도 훨신 좋아지고 불필요한 커넥션을 줄일수 있었다.

```
private class GetMapDataThread extends Thread {

	private Handler tHandler;

	private Double lat, lng;
	private String searchType;
	private String searchRange;

	public GetMapDataThread(Handler tHandler) {
		this.tHandler = tHandler;
	}

	public GetMapDataThread(Handler tHandler, Double lat, Double lng,
			String searhType, String searchRange) {
```

```
            this(tHandler); //스레드 처리 완료후 지도에 가져온 지점정보를 가지고 마커를 찍
어줄 핸들러
            this.lat = lat; //위도
            this.lng = lng; //경도
            this.searchType = searhType; //검색조건 (0 : 지점, 1: ATM)
            this.searchRange = searchRange; //검색범위 단위는 m(미터)이다.
      }

      @Override
      public void run() { //스레드 실행~

            Bundle bundle = new Bundle();

            try {
                  //전역변수로 선언한 지점 리스트를 준비한다. BranchInfoDTO 는 도메인이다.
                  brList = new ArrayList<BranchInfoDTO>();
                  brList = gdA.getMapData(lat.toString(), lng.toString(),
                              searchType, searchRange);
                  //gdA 클래스는 HTTP 통신을 해서 지점정보를 가져오는 클래스이다.
                  //여기서는 설명하지 않았다. onCreate 에서 생성했다.

                  bundle.putBoolean("SUCCESS_KEY", true); //성공하면 번들에 성공메세
지 셋팅

            } catch (Exception e) {
                  ...

                  bundle.putBoolean("SUCCESS_KEY", false); //실패하면 false 이다.
                  // ignore

            } finally {
                  try {
                        Message msg = tHandler.obtainMessage();
                        msg.setData(bundle);
                        tHandler.sendMessage(msg); //핸들러에 메세지를 보낸다.

                        interrupt();

                  } catch (Exception e) {
                        // ignore
                  }
            }

      }
}
```

스레드에서 HTTP 통신을 통하여 가져온 지점정보를 가지고 지도에 지점 마커들을 찍어주고 오버레이에 추가하는 핸들러이다.

```
final Handler getMapdataHandler = new Handler() {
      public void handleMessage(Message msg) {

            if (msg.getData().getBoolean("SUCCESS_KEY")) {  // HTTP 통신이 성공적으
로 이루어졌을때.

                  // draw branches
                  Drawable branchMarker;

                  int markerType = 0;
```

```
                    if ("0".equals(searchType)) { //검색조건에따라 마커이미지를 지점,ATM
중에 선택
                          markerType = R.drawable.icon_branch;
                    } else if ("1".equals(searchType)) {
                          markerType = R.drawable.icon_atm;
                    }

                    branchMarker = getResources().getDrawable(markerType);

                    branchMarker.setBounds(0, 0, branchMarker.getIntrinsicWidth(),
                                branchMarker.getIntrinsicHeight());

                    Double lat, lng;

                    //지점 마커들을 그려줄 오버레이를 준비한다.
                    overlayBranch = new LocationItemizedOverlay(branchMarker);
                    overlayBranch.mPopulate();

                    StringBuilder sb;
                    //반복문을 돌면서 마커들을 오버레이에 추가한다.
                    //나중에 마커를 눌렀을때 다이얼로그에 지점 정보를 보여주기위해 스니펫에 몇가
지 정보를
                    //string 으로 전달한다.

                    for (BranchInfoDTO d : brList) {

                          lat = Double.parseDouble(d.getYCord()) * 1E6;
                          lng = Double.parseDouble(d.getXCord()) * 1E6;
                          GeoPoint branchGeoPoint = new GeoPoint(lat.intValue(),
                                      lng.intValue());

                          sb = new StringBuilder();
                          sb.append(d.getBussBrNm()).append(";")
                                      .append(d.getBussBrTelNo()).append(";")
                                      .append(d.getBussBrAdr()).append(";")
                                      .append(d.getTrscDrtm()).append(";")
                                      .append(d.getBussBrAdr2());

                          // Create new overlay with marker at geoPoint
                          OverlayItem overlayItem = new OverlayItem
(branchGeoPoint,
                                      "branch", sb.toString());
                          overlayBranch.addOverlay(overlayItem);
                    }

              }

              //마커 찍은것이 없으면 오류 메세지를 토스트로 보여준다.
               if (overlayBranch.size() < 1){
                     Toast.makeText(getApplicationContext(),
                                "검색결과가 없거나 통신장애 입니다.\n'메뉴'버튼을 눌
러 조건을 변경하여 다시 검색해 주세요.",
                                Toast.LENGTH_LONG).show();
               }

              //지점 오버레이를 맵뷰 오버레이에 최종적으로 추가해준다.
              if (overlayBranch != null) {
                    mapView.getOverlays().add(overlayBranch);
                    mapView.postInvalidate();
              }

        };
};
```

토스트 메세지로 현재 주소와 위도,경도를 잠시 표시해주는 메소드.

```
private String showNowHere(double lat, double lng , boolean showOption){
	StringBuilder geoString = new StringBuilder();
	try {
		Geocoder goecoder = new Geocoder(getApplicationContext(),
				Locale.getDefault());

		Address adr = goecoder.getFromLocation(lat,
				lng, 1).get(0);

		if (adr.getLocality() != null) geoString.append(adr.getLocality
()).append(" ");
		if (adr.getThoroughfare() != null) geoString.append
(adr.getThoroughfare());
		if (!"".equals(geoString.toString())) geoString.append("\n\n");

	} catch (Exception e) { }

	geoString.append("위도 : ").append(lat).append(" ,경도 : ").append(lng);

	if (showOption){
		Toast.makeText(getApplicationContext(), geoString.toString(),
				Toast.LENGTH_SHORT).show();
	}

	return geoString.toString();
}
```

캡춰 화면에서 ‘서울특별시 신천동’과 위,경도가 떠있는 토스트이다.
그런데 ‘송파구’ 를 어떻게 가져오는지 모르겠다 -_-;;

이 다음은 화면에서 터치 이벤트를 받아올 오버레이이다.
맵뷰에서 특정지점을 누르고 있으면 현재위치가 아닌 특정지점을 기준으로 지점정보를 검색해오려고 만든 오버레이인데 길게 누르는 이벤트를 받아오는 방식이 좀 어거지이다.
분명 이부분은 개선이 되어야 할것이다.

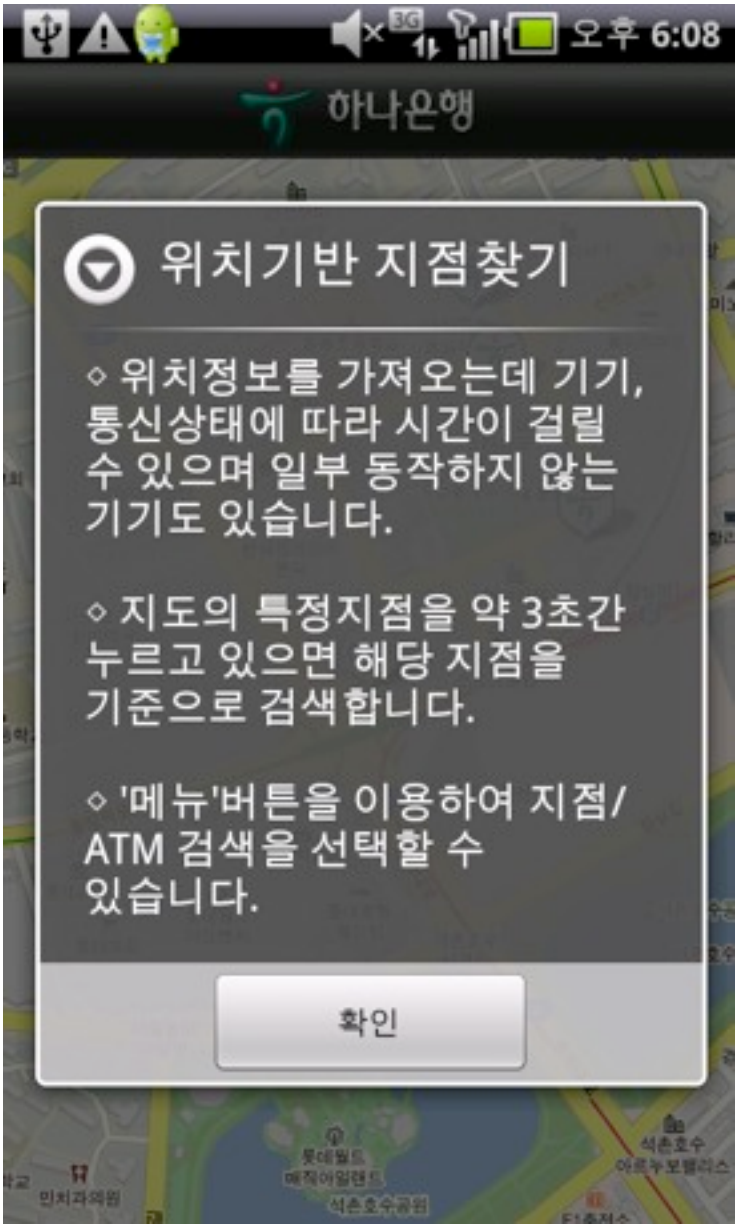
```
public class MapTouchDetectorOverlay extends Overlay implements
            OnGestureListener {
      private GestureDetector gestureDetector;

      //onTouchEvent 의 ACTION_DOWN 등을 가지고 직접 처리 하지 않고
      //제스처들을 쉽게 캐치할수있는 리스너이다.
      private OnGestureListener onGestureListener;

      private static final long LOOOOONG_PRESS_MILLI_SEC = 1500; // 1.5초정도를 길
게누름으로 인식한다.
```

```
	// for touch timer
	private Handler mHandler;
	private long touchStartTime;
	private long longPressTime;
	private MotionEvent globalEvent;

	//생성자
	public MapTouchDetectorOverlay() {
		gestureDetector = new GestureDetector(this);
		init();
	}

	public MapTouchDetectorOverlay(OnGestureListener onGestureListener) {
		this();
		setOnGestureListener(onGestureListener);
		init();
	}

	//생성자들이 호출할 초기화 함수
	private void init() {
		mHandler = new Handler();
		globalEvent = null;
	}

	//길게누름을 감지할 스레드
	private Runnable looongPressDetector = new Runnable() {
		public void run() {
			//화면을 누르고 있던 시간
			long touchHoldTime = longPressTime - touchStartTime;
			if ((globalEvent != null)
					&& (touchHoldTime > (LOOOOONG_PRESS_MILLI_SEC -
200))) { //조건중에 200ms 를 빼고 검사하는것은 기기마다 성능이 달라서 약간의 여유를 준것이다.
				Log.d(LOG_TAG, "looooooong press detected!");

				float x = globalEvent.getX();
				float y = globalEvent.getY(); //화면에서 눌려있던 지점을 받아
온다.

				GeoPoint p = mapView.getProjection().fromPixels((int) x,
						(int) y); //눌려있던 지점을 위도 경도로 바꿔준다.

				Location selectedLocation = new Location
(currentLocation);

				selectedLocation.setLatitude((p.getLatitudeE6() / 1E6));
				selectedLocation.setLongitude((p.getLongitudeE6() /
1E6));
				currentLocation = selectedLocation;

				locM.removeUpdates(locL); //현재위치 리스너를 잠시 없애버린다.
				udateOverlay(currentLocation); //지점 재검색 및 마커 다시 표
시

				showNowHere((p.getLatitudeE6() / 1E6) ,
(p.getLongitudeE6() / 1E6) , true);
			}
		}
	};

	@Override
	public boolean onTouchEvent(MotionEvent event, MapView mapView) {
		if (gestureDetector.onTouchEvent(event)) {
			return true;
		}
```

```
			onLongPress(event);
			return false;
		}

		@Override
		public boolean onDown(MotionEvent e) {
			if (onGestureListener != null) {
				return onGestureListener.onDown(e);
			} else {
				// start timer
				touchStartTime = System.currentTimeMillis();
				mHandler.postDelayed(looongPressDetector,
						LOOOOONG_PRESS_MILLI_SEC);
				//1.5초 있다가 길게누름을 체크해본다.
			}

			return false;
		}

		@Override
		public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
				float velocityY) {
			if (onGestureListener != null) {
				return onGestureListener.onFling(e1, e2, velocityX,
velocityY);
			}
			return false;
		}

		@Override
		public void onLongPress(MotionEvent e) {
			if (onGestureListener != null) {
				onGestureListener.onLongPress(e);
			}

			//화면을 누르고 있으면 onLongPress 가 호출되는데 호출될때마다 체크할 시간을 변수
에 넣는다.
			//이부분이 퍼포먼스 하락에 영향을 줄 것 같다.
			globalEvent = e;
			longPressTime = System.currentTimeMillis();

		}

		@Override
		public boolean onScroll(MotionEvent e1, MotionEvent e2,
				float distanceX, float distanceY) {
			if (onGestureListener != null) {
				onGestureListener.onScroll(e1, e2, distanceX, distanceY);
			}
			return false;
		}

		@Override
		public void onShowPress(MotionEvent e) {
			if (onGestureListener != null) {
				onGestureListener.onShowPress(e);
			}
		}

		@Override
		public boolean onSingleTapUp(MotionEvent e) {
			if (onGestureListener != null) {
				onGestureListener.onSingleTapUp(e);
			}
```

```
            return false;
        }

        public boolean isLongpressEnabled() {
            return gestureDetector.isLongpressEnabled();
        }

        public void setIsLongpressEnabled(boolean isLongpressEnabled) {
            gestureDetector.setIsLongpressEnabled(isLongpressEnabled);
        }

        public OnGestureListener getOnGestureListener() {
            return onGestureListener;
        }

        public void setOnGestureListener(OnGestureListener onGestureListener) {
            this.onGestureListener = onGestureListener;
        }

}
```

완성된 지점찾기의 동작모습.
액티비티를 실행하게 되면 다음과 같이 작동한다.


오후 6:08
하나은행
위치기반 지점찾기
◇ 위치정보를 가져오는데 기기, 통신상태에 따라 시간이 걸릴 수 있으며 일부 동작하지 않는 기기도 있습니다.
◇ 지도의 특정지점을 약 3초간 누르고 있으면 해당 지점을 기준으로 검색합니다.
◇ '메뉴'버튼을 이용하여 지점/ATM 검색을 선택할 수 있습니다.
확인


오후 6:09
하나은행
서울특별시 신천동
위도 : 37.5137842 ,경도 : 127.1004029


오후 6:09
하나은행
잠실대교
올림픽대로
잠실대교남단
로즈버드
잠실고등학교
성내역
신천장미
2차APT
잠실세아올
림고
잠실파크
2단지APT
신천장미
1차APT
잠실중학교
한화갤러리아
본사
잠실주공
5단지APT
현위치
현대타워
잠실역
한솔건설 본사
송파구청
잠실5동
우체국
롯데월드
어드벤처
롯데물산
미니스탑
석촌호수공원
대우레이크
빌드APT
송파대로
롯데월드
매직아일랜드
석촌호수
아르누보팰리스
잠실레이크
팰리스APT
삼학사길
헤이리마트
GS25
GS주유소
삼전도비
백제고분로


오후 6:09
하나은행
잠실대교
올림픽대로
잠실대교남단
88
잠실고등학교
성내역
신천장미
2차APT
잠실세아올
림고
잠실파크
2단지APT
신천장미
1차APT
잠실중학교
한화갤러리아
본사
잠실주공
5단지APT
현위치
현대타워
잠실역
한솔건설 본사
송파구청
잠실5동
우체국
롯데월드
어드벤처
롯데물산
미니스탑
석촌호수공원
대우레이크
빌드APT
송파대로
서울특별시 신천동
위도 : 37.5137842 ,경도 : 127.1004029
해미리마트
백제고분로
GS주유소
삼전도비


오후 6:09
하나은행
잠실대교
올림픽대로
잠실고등학교
잠실레이크팰리스
레이크팰리스 상가 내
ATM : 07:00~24:00
대표전화 : 02-3432-1111
확인
전화걸기
롯데월드
매직아일랜드
석촌호수
아르누보팰리스
잠실레이크
팰리스APT
삼전도비


오후 6:10
하나은행
로즈버드
잠실대교
잠실고등학교
올림픽대로
성내역
잠실대교남단
신천장미
2차APT
잠실파크
2단지APT
잠실새아울
금고
신천장미
1차APT
잠실중학교
한화갤러리아
본사
잠실주공
5단지APT
현위치
현대타워
잠실역
한솔건설 본사
송파구청
잠실5동
우체국
롯데월드
어드벤처
롯데몰산
미니스탑
석촌호수공원
대우레이크
빌드APT
송파대로
롯데월드
매직아일랜드
석촌호수
아르누보팰리스
잠실레이크
팰리스APT
삼학사길
지점찾기
ATM찾기
현위치


오후 6:10
하나은행
사근동
중곡1동
송정동
능동
동부간선도로
성수2가3동
광진대교
천호대교
광진구
구의3동
강변북로
동2로
올림픽대로
영동대교
청담대교
잠실대교
현위치
강남구
삼성2동
삼성1동
잠실7동
분당정자간고속
수서
일원1동
일원2동
개포2동
수서동
문정2동
양재대로
일원본동
도곡2동
개포1동
개포4동
양재2동
세곡동


오후 6:11
하나은행
성수2가3동
응봉산
동부간선도로
광진구
성수대교
동호대교
강변북로
영동대교
청담대교
신사동
논현2동
논현1동
잠실7동
반포
올림픽대로
분당장지간고속
역삼1동
서초2동
도곡1동
개포2동
임원1동
서초
서초구
도곡2동
개포1동
개포4동
양재2동
경부고속
양재대로
임원본동


오후 6:12
하나은행
서울특별시 개봉동
위도 : 37.489465 ,경도 : 126.85745