

Daftar Isi

apa aja isi bukunya

| | |
|----------------------------------|-----------|
| Edition? | 2 |
| Daftar Isi | 4 |
| Revisi Buku | 7 |
| Persiapan | 8 |
| Spesifikasi Aplikasi | 8 |
| Perangkat lunak yang dibutuhkan | 9 |
| Mengirim SMS | 11 |
| Bikin Project Android | 11 |
| Gak perlu menu options | 13 |
| Jangan gunakan Android Emulator | 14 |
| Jalankan aplikasi-nya | 15 |
| Cara kerja aplikasi | 17 |
| Bikin SMS Gateway WebServer | 18 |
| Membuat HttpRequestHandler | 21 |
| Menambahkan Handler | 21 |
| Format isi pesan HTTP request | 22 |
| Menangkap isi pesan HTTP request | 23 |
| Mengirim SMS | 23 |
| Menambah uses-permission | 25 |
| Menjalankan SMS server | 26 |
| Client API | 30 |
| Membuat HotSpot | 30 |
| Mengecek IP Address | 31 |
| Mengirim SMS Menggunakan POSTMan | 32 |
| Mengirim SMS Menggunakan Java | 35 |
| Mengirim SMS dari NodeJS | 36 |
| Mengirim SMS Menggunakan PHP | 36 |
| Mengirim SMS Menggunakan Ruby | 37 |
| What's Next? | 38 |
| Premium Edition | 40 |

| | |
|--|-----------|
| Rombak Ulang App | 41 |
| Kenapa HTTP Request Tidak Cocok? | 41 |
| Memilih Framework WebSocket | 42 |
| Menambah Library Java-WebSocket | 43 |
| Membuat SmsGatewayContainer | 43 |
| Membuat SmsGatewayServer | 44 |
| Format Pesan Request | 46 |
| Mengirim SMS | 47 |
| Menjalankan SMS Gateway Server | 47 |
| Menambah Permission | 48 |
| Client API | 50 |
| Menambah Library Java-WebSocket | 50 |
| Membuat WebSocketClient | 50 |
| Menambah Library Google GSON | 52 |
| Mengirim Pesan ke Server | 52 |
| Menerima Laporan | 54 |
| Membuat metode send() di SmsGatewayContainer | 54 |
| Menambahkan PendingIntent | 55 |
| Menambahkan PendingIntent dari MainActivity | 56 |
| Membuat Receiver | 57 |
| Menerima SMS | 60 |
| Membuat BroadcastReceiver | 60 |
| Meregistrasikan BroadcastReceiver | 62 |
| Menambah uses-permission RECEIVE_SMS | 62 |
| Source Code | 64 |
| Ultimate Edition | 66 |
| Revisi Dikit | 67 |
| Revisi onMessage() SmsGatewayServer | 68 |
| Revisi Laporan Pengiriman SMS | 69 |
| Revisi SmsGatewayReceiver | 71 |
| Broadcast Message | 72 |
| Format pesan broadcast message | 72 |
| Mengubah onMessage SmsGatewayServer | 73 |
| Fitur Apa Lagi? | 75 |
| Aplikasi Desktop | 76 |
| Membuat Project | 76 |

| | |
|--|------------|
| Menambahkan Library | 78 |
| Membuat Form Aplikasi | 78 |
| Mendesain Tampilan Form Aplikasi | 79 |
| Membuat WebSocketClient | 81 |
| Membuat WebSocketListener | 82 |
| Menambahkan Listener ke SmsGatewayClient | 83 |
| Implementasi Listener di AppForm | 85 |
| Menjalankan SmsGatewayClient | 89 |
| Menambah Aksi tombol Send SMS | 89 |
| Bagaimana dengan Broadcast Message? | 90 |
| Mengaju Aplikasi | 91 |
| Aplikasi Web | 95 |
| Membuat Project Web | 96 |
| Membuat Form Web | 96 |
| Menambah Aksi Send SMS | 99 |
| Implementasi onmessage WebSocket Client | 99 |
| Broadcast Message | 101 |
| Menguji Aplikasi | 103 |
| In The END | 106 |
| Source Code | 107 |
| Penulis | 108 |

Persiapan

yang perlu dilakukan sebelum bertarung

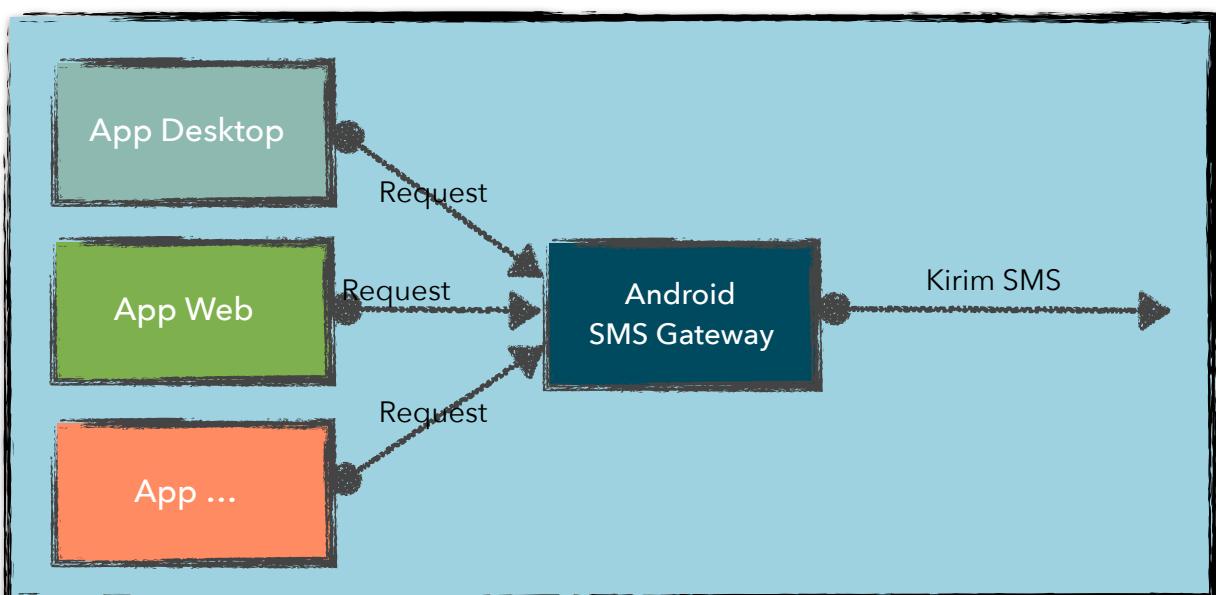
Buku ini adalah buku praktek, bukan buku teori, hampir 90% isi buku ini adalah praktek implementasi membuat aplikasi. Saya hanya akan membahas sedikit soal teorinya, jika memang diperlukan saja.

Spesifikasi Aplikasi

Sebelum mulai membuat aplikasi, kita akan tentukan terlebih dahulu spesifikasi aplikasi Android SMS Gateway yang akan kita buat.

Aplikasi Android SMS Gateway yang akan kita buat adalah sebuah aplikasi yang menjadi jembatan penghubung antara aplikasi lain (misal desktop, web, mobile, dll) agar memiliki kemampuan untuk mengirim SMS.

Aplikasi Android SMS Gateway bukanlah aplikasi pengolah pesan SMS, aplikasi seperti itu sudah ada di tiap hp Android. Tujuan utama nya adalah menjadi **PENGHUBUNG**.



Fitur Apa saja yang akan dimiliki oleh aplikasi Android SMS Gateway yang akan kita buat?

- ▶ Bisa mengirim SMS pastinya
- ▶ Bisa diakses oleh teknologi apapun (PHP, Java, Ruby, NodeJS, dan lain-lain)
- ▶ Client yang mengakses boleh menggunakan sistem operasi apapun (Windows, Linux dan Mac)
- ▶ Perintah ke aplikasi tidak dikirim menggunakan perangkat pihak ke-3 (seperti GAMMU), bahkan tidak menggunakan AT COMMAND yang ribetnya minta ampun.
- ▶ Komputer client tidak perlu terkoneksi langsung ke aplikasi (melalui kabel USB), hanya cukup dalam satu jaringan yang sama (wifi)

Khusus untuk **PREMIUM** dan **ULTIMATE EDITION**, aplikasi yang akan dibuat juga memiliki fitur :

- Bisa memberi tahu laporan SMS terkirim
- Bisa memberi tahu jika ada SMS masuk secara REAL-TIME
- Dapat mengirim langsung ke beberapa nomor tujuan dalam satu request

Perangkat lunak yang dibutuhkan

Untuk belajar dan praktik buku ini, saya akan gunakan semua perangkat lunak yang dapat di download secara gratis, dengan tujuan supaya tidak ada alasan untuk tidak bisa belajar karena gak sanggup beli perangkat lunaknya :D

Apa saja perangkat lunak yang akan digunakan dalam buku ini?

- ▶ **Java Development Kit**, pada buku ini, kita akan praktekan menggunakan bahasa pemrograman Java. Wah saya gak bisa Java? Oke, tinggal Anda belajar Java dulu :) Karena untuk membuat aplikasi Android, Anda sangat dianjurkan mengerti pemrograman Java.
- ▶ **Android Studio**. IDE yang akan kita gunakan dalam buku ini adalah Android Studio (berbasis IntelliJ IDEA), namun jika Anda lebih suka menggunakan Android Development Tool (berbasis Eclipse), Anda tetap bisa mengikuti tutorial di buku ini.
- ▶ **Koneksi Internet**. Untuk download dependency secara langsung, sehingga tidak perlu download manual. Capek kalo harus download manual :P
- ▶ Cukup!

Tidak banyak yang harus kita siapkan, hanya 3 saja, JDK, IDE (Android Studio atau ADT) dan koneksi Internet. Kalo 3 hal itu sudah siap, sekarang saatnya kita bertempur!

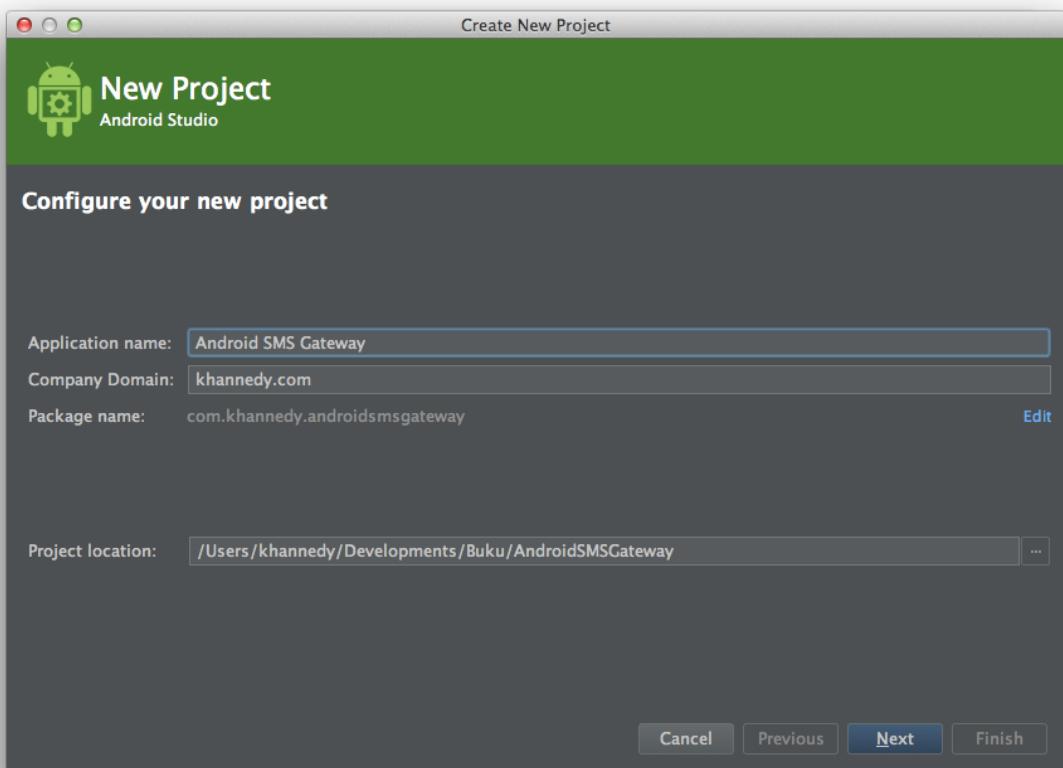
Mengirim SMS

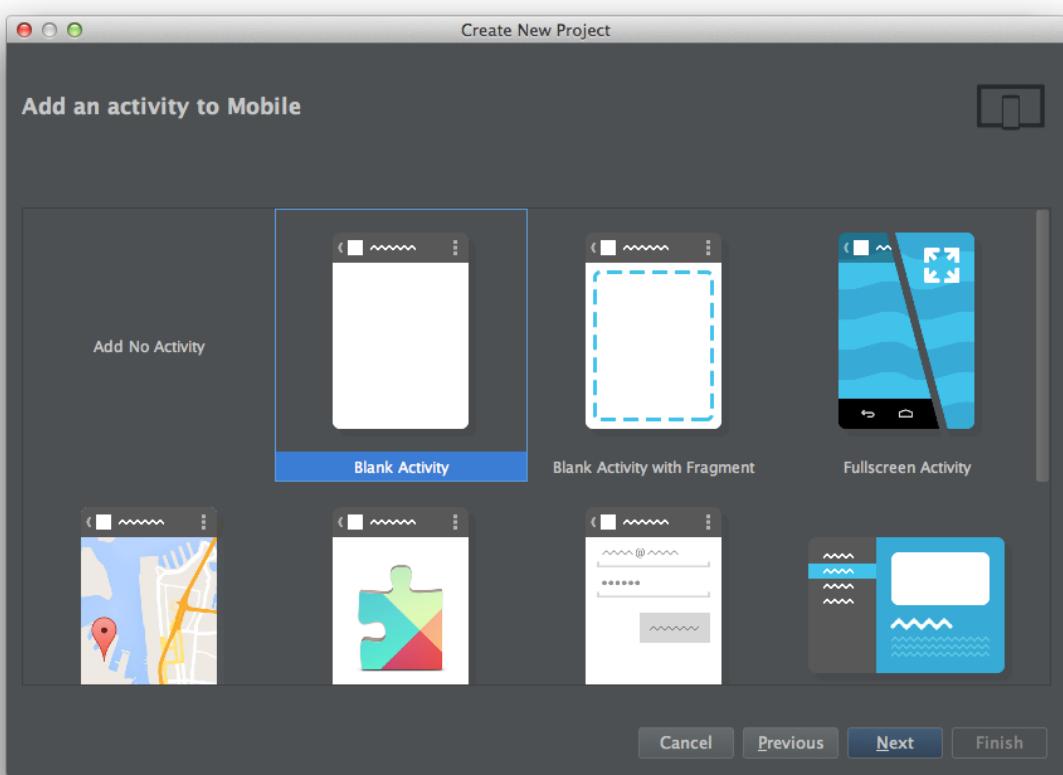
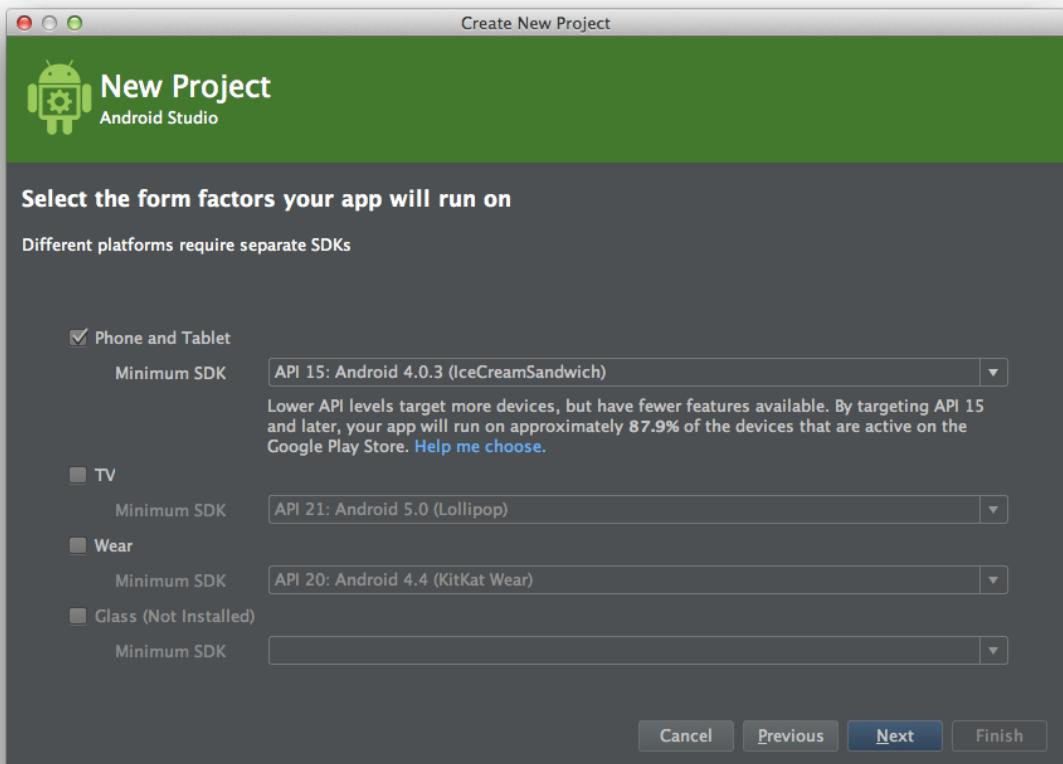
bikin app Android yang bisa ngirim sms

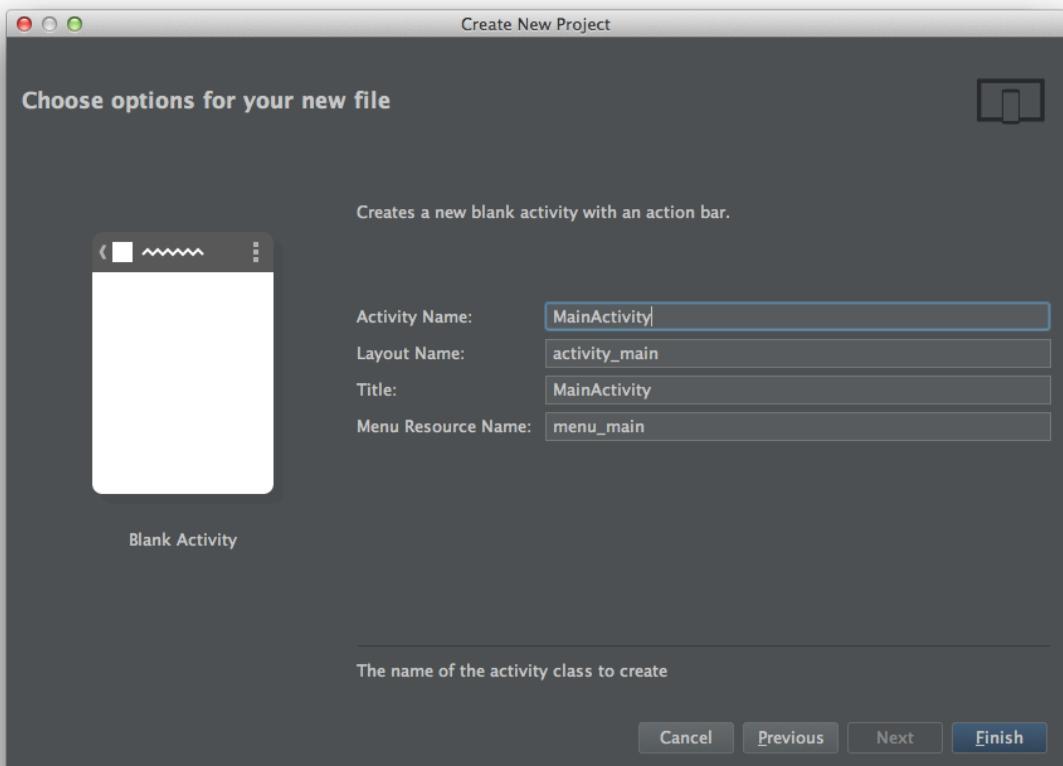
Fitur yang akan kita buat di Aplikasi Android SMS Gateway nya yang pasti adalah mampu mengirim SMS, kalo enggak bisa mengirim SMS, ya sama aja boonk namanya :D

Bikin Project Android

Seperti yang sudah dibahas sebelumnya, disini kita akan menggunakan Android Studio, kalo Anda ingin menggunakan ADT, itu terserah, sama saja. Sekarang silahkan buat project Android menggunakan Android Studio seperti pada tahapan yang terlihat pada gambar.







Buat sebuah project dengan template Blank Activity, sengaja menggunakan Blank Activity supaya secara otomatis dibuatkan sebuah layout **activity_main.xml**. Pada aplikasi Android SMS Gateway yang akan kita buat, tidak akan terlalu fokus pada UI (User Interfaces), namun lebih fokus ke kemampuan aplikasinya.

Gak perlu menu options

Secara default, saat kita buat sebuah project Blank Activity, akan dibuatkan sebuah option menu. Kita tidak perlu options menu tersebut, jadi kita bisa hapus option menu nya.

Silahkan buka kode **MainActivity.java** dan hapus metode **onCreateOptionsMenu** dan **onOptionsItemSelected** sehingga kodennya hanya seperti dibawah ini.

```
package com.khannedy.androidmsgateway;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

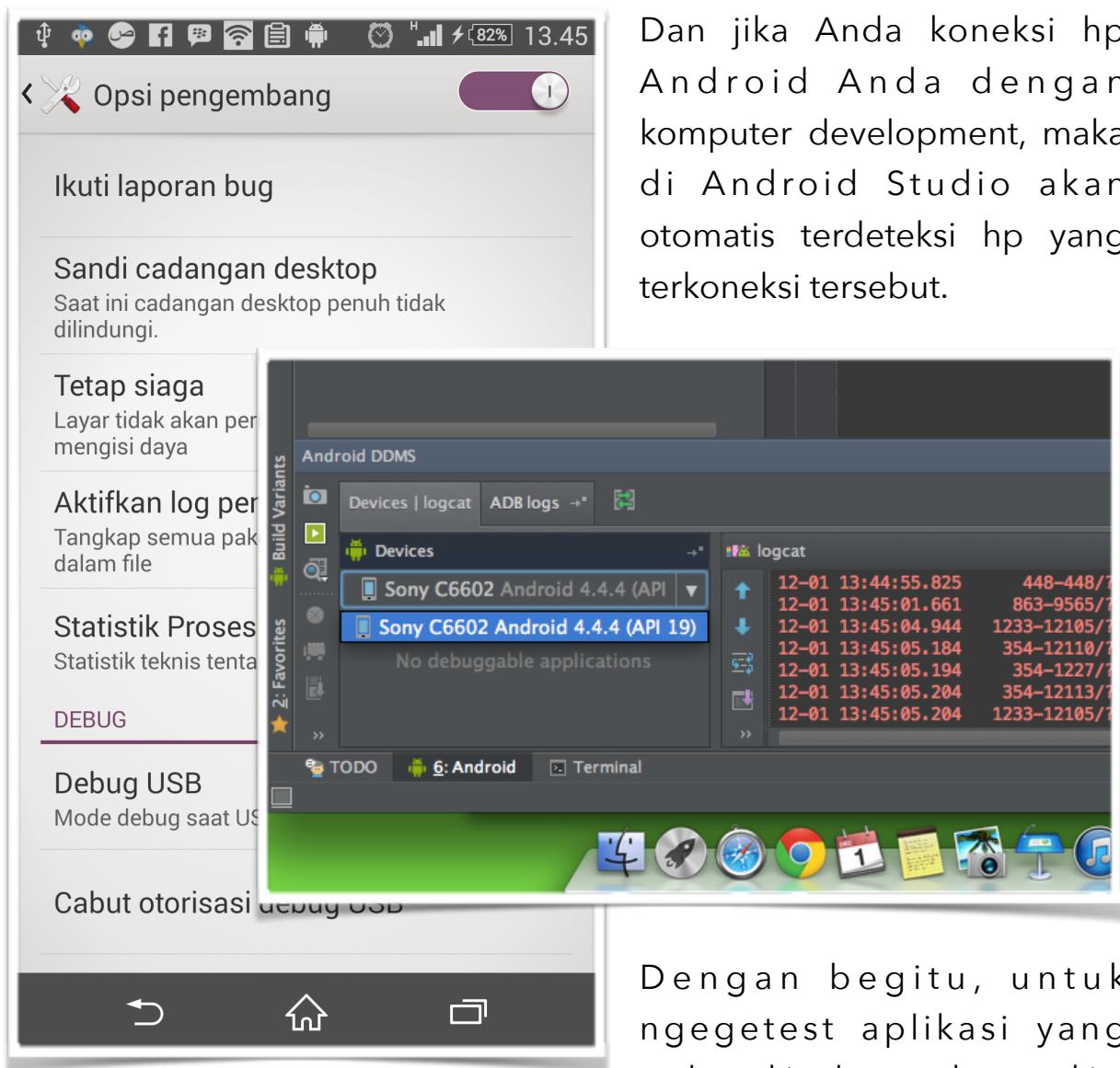
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

Jangan gunakan Android Emulator

Saya sangat tidak suka menggunakan Android Emulator, why? karena sangat lambat sekali. Jadi saran saya, lebih baik gunakan device Android aslinya. Di tutorial ini saya menggunakan hp Sony Xperia Z.

Jika kita ingin gunakan device Android kita untuk develop aplikasi, pastikan opsi pengembangnya aktif. Secara default opsi pengembanya tidaklah aktif. Cara mengaktifkan opsi pengembang biasanya berbeda tiap merek HP, jika di Sony cukup masuk menu **Pengaturan -> Opsi Pengembang**. Lalu aktifkan.



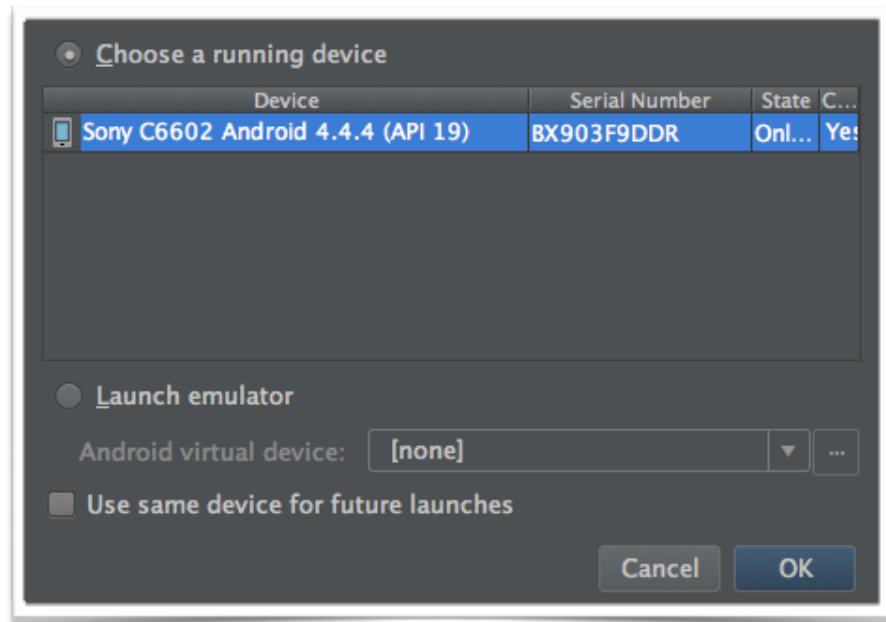
Dengan begitu, untuk ngegetest aplikasi yang sedang kita buat sekarang kita bisa langsung publish ke device asli, tanpa menggunakan Emulator

Memang kenapa harus device asli? Ya karena kita akan membuat aplikasi SMS gateway, sedangkan kalo menggunakan emulator kita tidak bisa melakukan pengiriman sms :D

Jalankan aplikasi-nya

Untuk menjalankan aplikasi yang telah kita buat. Lho emang udah kita buat? Belum sih, tapi sebelum lanjut, perlu tau caranya gimana cara jalanin aplikasi di device Android nya.

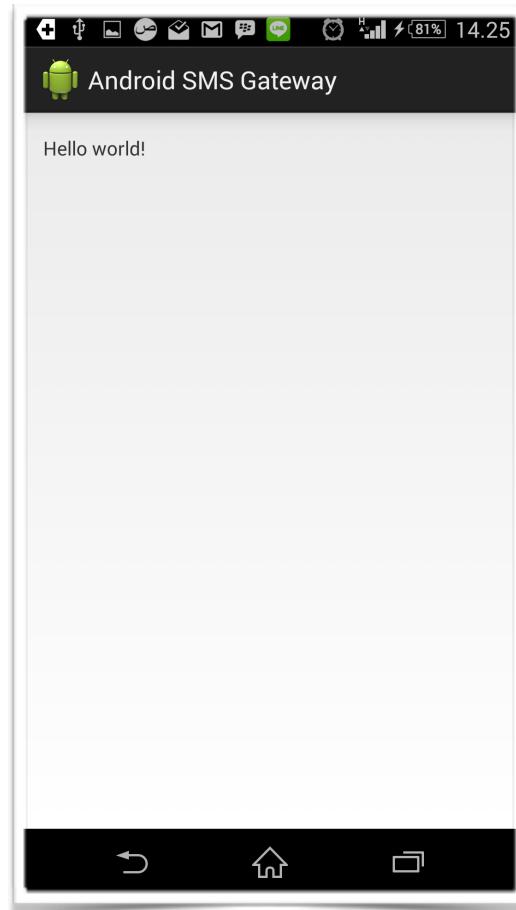
Silahkan pilih menu **Run -> Run 'app'**. Tunggu sampai proses build selesai, ketika proses build selesai, kita akan diminta untuk memilih tujuan run aplikasinya.



Silahkan pilih Device Android yang Anda gunakan saat praktik buku ini. Berikut adalah contoh hasil dari aplikasi yang sudah kita buat.

Hanya sebuah aplikasi dengan label "Hello world!". Karena memang sampai sekarang kita belum membuat apapun.

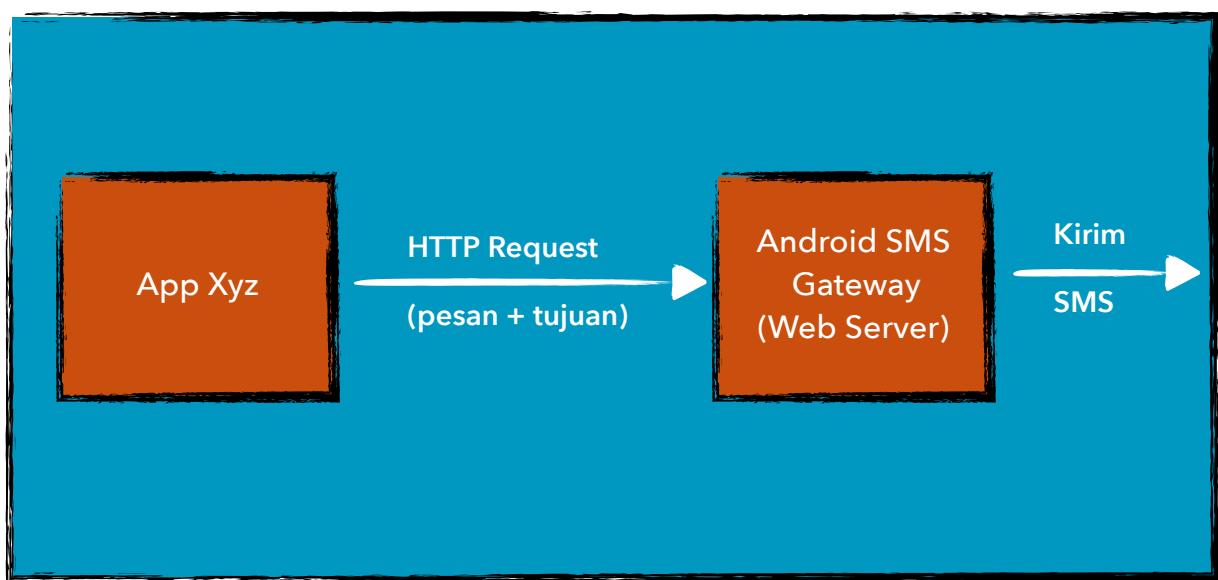
Sekarang kita kembali ke topik yang akan kita pecahkan (kaya detektif ajah), kita lanjutkan membuat Android SMS Gateway.



Cara kerja aplikasi

Sebelum kita buat, kita akan bahas dulu bagaimana cara kerja aplikasinya. Bagaimana cara aplikasi menangkap request dari aplikasi lain untuk mengirim SMS.

Hal yang paling mudah adalah, kita akan buat sebuah WEB SERVER yang berjalan diatas Android. Wow! Web Server? Yup benar, saya serius, dua-rius malah :D Nanti web server-nya akan menangkap HTTP Request, lalu membaca isi data nya (pesan sms + nomor tujuan), setelah membaca isi datanya, baru web servernya akan mengirim sms ke nomor tujuan.



Idenya memang cukup sederhana. Namun pertanyaannya, bagaimana cara menjalankan web server di Android? Bukankah web server adalah aplikasi yang perlu dijalankan di komputer server?

Haha, tenang saja, kita tidak akan menggunakan web server? Maksudnya? Yup kita tidak akan menggunakan web server apapun, seperti Apache HTTPD, Apache Tomcat, NGINX atau Jetty, tapi **kita akan buat web servernya secara manual!** Serius? Yup tentu saja :D

Bikin SMS Gateway WebServer

Yakin kita akan bikin web sever manual? Yup, kenapa tidak? Jangan terlalu berpikir kalo sebuah web server itu adalah aplikasi yang berat yang hanya bisa jalan di komputer. Sebenarnya aplikasi web server itu tidak berbeda dengan aplikasi network yang menggunakan socket. Yang membedakan adalah web server memiliki standard request dan response menggunakan standard HTTP.

Jadi sebenarnya yang hanya akan kita buat hanyalah aplikasi server socket. Lantas kenapa gak server socket tok aja? Kenapa harus menggunakan standard HTTP? Hal ini karena saya berpikir, jika menggunakan server socket, maka client harus implementasi manual menggunakan socket client, sedangkan kalo menggunakan standard HTTP, client bisa menggunakan library yang begitu banyak untuk HTTP Client. Hal ini menjadikan lebih mudah saat integrasi. Dan yang tak kalah penting, jika menggunakan standard HTTP, kita juga bisa mengetestnya cukup menggunakan web browser :D

Sekarang back to kode, mari kita buat sebuah kelas baru bernama **SmsGatewayServer.java**.

```
package com.khannedy.androidsmsgateway;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayServer {

    public SmsGatewayServer(int port) { }

    public void start() { }

    public void stop() { }
}
```

- ▶ Pada konstruktor kelas **SmsGatewayServer** terdapat parameter **int port**. Parameter tersebut nanti akan kita gunakan sebagai penentuan web server akan berjalan di port berapa.
- ▶ Metode **start()** akan kita gunakan untuk menjalankan web server SMS Gateway
- ▶ Metode **stop()** akan kita gunakan untuk menghentikan web server SMS Gateway.

Sampai sini masih mudah kan? Oke sekarang tinggal kita implementasi satu per satu dimulai dari implementasi konstruktor SmsGatewayServer nya.

```
package com.khannedy.androidsmsgateway;

import org.apache.http.*;
import org.apache.http.impl.*;
import org.apache.http.params.*;
import org.apache.http.protocol.*;

import java.io.*;
import java.net.*;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayServer {

    private int port;
    private HttpService httpService;
    private ServerSocket serverSocket;
    private HttpContext httpContext;

    public SmsGatewayServer(int port) {
        this.port = port;
        httpService = new HttpService(new BasicHttpProcessor(),
            new DefaultConnectionReuseStrategy(),
            new DefaultHttpResponseFactory());
        httpContext = new BasicHttpContext();
    }
}
```

Pada kode diatas, kita memanfaatkan Apache HTTP Client yang secara default sudah terdapat di library Android. Apache HTTP Client yang akan kita gunakan nanti bertugas untuk mengubah request dan response dari server socket yang kita buat agar sesuai dengan spek HTTP, dengan begitu kita tidak perlu membuat response HTTP secara manual.

Sekarang kita implementasikan metode start() dimana dalam metode start() kita akan jalankan ServerSocket, lalu terima setiap request yang masuk, handle sebagai HTTP Request, lalu close konesinya setelah itu.

```
public void start() throws IOException, HttpException {
    // membuat server socket berdasarkan port
    serverSocket = new ServerSocket(port);

    while (true) {
        // terima socket client jika ada request masuk
        Socket socket = serverSocket.accept();

        // handle request sebagai HTTP Request
        DefaultHttpServerConnection sCon = new DefaultHttpServerConnection();
        sCon.bind(socket, new BasicHttpParams());
        httpService.handleRequest(sCon, httpContext);

        // close koneksi client
        socket.close();
    }
}
```

Terakhir adalah implementasikan metode stop(), metode ini sangat simple, hanya menutup ServerSocket saja.

```
public void stop() throws IOException {
    serverSocket.close();
}
```

Membuat **HttpRequestHandler**

Karena kita menggunakan Apache HTTP Client sebagai library untuk menghandle HTTP request nya, maka kita perlu membuat sebuah handler. Handler disini merupakan kode yang nanti panggil oleh Apache HTTP Client jika ada HTTP request yang masuk ke Server Socket.

Untuk membuat sebuah handler, kita cukup membuat sebuah kelas turunan dari **HttpRequestHandler**. Misal kita beri nama dengan nama **SmsGatewayHandler**

```
package com.khannedy.androidsmsgateway;

import android.util.Log;
import org.apache.http.*;
import org.apache.http.protocol.*;
import org.apache.http.util.*;
import org.json.*;

import java.io.IOException;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayHandler implements HttpRequestHandler {

    @Override
    public void handle(HttpRequest httpRequest, HttpResponse httpResponse,
                      HttpContext httpContext) throws HttpException, IOException {
    }
}
```

Tahan dulu, sebelum kita implementasikan handlernya, terlebih dahulu kita harus memberi tahu ke Apache HTTP Client nya bahwa kita telah membuat sebuah handler.

Menambahkan Handler

Untuk menambahkan handler, kita perlu menambahkannya di `HttpService` yang kita buat di `SmsGatewayServer`. Tambahkan menggunakan `HttpRequestHandlerRegistry`, sehingga sekarang konstruktor `SmsGatewayServer` nya menjadi seperti berikut.

```
public SmsGatewayServer(int port) {
    this.port = port;
    httpService = new HttpService(new BasicHttpProcessor(),
        new DefaultConnectionReuseStrategy(),
        new DefaultHttpResponseFactory());
    httpContext = new BasicHttpContext();

    // tambahkan handler
    HttpRequestHandlerRegistry registry = new HttpRequestHandlerRegistry();
    registry.register("*", new SmsGatewayHandler());
    httpService.setHandlerResolver(registry);
}
```

Setelah `SmsGatewayHandler` kita registrasikan ke `HttpService`, sekarang baru kita implementasikan isi handler nya :D

Format isi pesan HTTP request

Sebelum menangkap isi pesan HTTP request, kita perlu menentukan format seperti apa nanti pesan yang akan kita terima. Supaya lebih mudah dingerti, kita akan buat format pesan-nya dalam bentuk String JSON seperti pada kode dibawah ini :

```
{
    "no" : "08111111111",
    "pesan" : "hello apa kabar bro!"
}
```

Atribut “no” berisikan nomor telepon tujuan, dan atribut “pesan” adalah isi pesan SMS.

Menangkap isi pesan HTTP request

Sekarang mari kita tangkap isi pesan String JSON yang terdapat pada HTTP request yang masuk ke handler. Ubah isi metode handle pada kelas SmsServerHandler-nya.

```
@Override
public void handle(HttpServletRequest httpRequest, HttpServletResponse httpResponse,
                     HttpContext httpContext) throws HttpException, IOException {

    if (httpRequest instanceof HttpEntityEnclosingRequest) {
        // HTTP request haruslah memiliki body
        try {
            HttpEntity entity = ((HttpEntityEnclosingRequest) httpRequest).getEntity();

            // convert String body menjadi JSON
            String body = EntityUtils.toString(entity);
            JSONObject object = new JSONObject(body);

            // ambil no tujuan dari json
            String no = object.getString("no");
            // ambil pesan dari json
            String pesan = object.getString("pesan");

            // TODO kirim SMS

        } catch (Exception ex) {
            Log.e(SmsGatewayHandler.class.getName(), ex.getMessage(), ex);
        }
    }
}
```

Sekarang kita telah menangkap isi pesan String JSON dan kita simpan dalam variabel no (untuk nomor tujuan) dan variabel pesan untuk isi pesan SMS. Saatnya kita kirim pesan SMS nya ke nomor tujuan tersebut.

Mengirim SMS

Dan akhirnya! Mengirim SMS!!!! Hehehehehe, panjang banget ya sebelum mencapai ke sini :D Yah itulah pengorbanan supaya nanti

saat kita bikin aplikasi yang akan menggunakan Android SMS Gateway ini, menjadi lebih mudah integrasinya :)

Sebenarnya mengirim SMS menggunakan Android itu sangat mudah, cukup 1 baris saja! Hah! Serius? Yup saya serius :D Cukup dengan perintah :

```
SmsManager.getDefault().sendTextMessage("no tujuan", null, "pesan sms", null, null);
```

Dan berikut adalah hasil akhir dari handler yang kita buat setelah ditambah kode untuk mengirim sms menggunakan SmsManager.

```
package com.khannedy.androidsmsgateway;

import android.util.Log;
import org.apache.http.entity.*;
import org.apache.http.*;
import org.apache.http.protocol.*;
import org.apache.http.util.*;
import org.json.*;

import java.io.IOException;
import android.telephony.SmsManager;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayHandler implements HttpRequestHandler {

    @Override
    public void handle(HttpRequest httpRequest, HttpResponse httpResponse,
                       HttpContext httpContext) throws HttpException, IOException {

        if (httpRequest instanceof HttpEntityEnclosingRequest) {
            // HTTP request haruslah memiliki body
            try {
                HttpEntity entity = ((HttpEntityEnclosingRequest) httpRequest).getEntity();

                // convert String body menjadi JSON
                String body = EntityUtils.toString(entity);
                JSONObject object = new JSONObject(body);

                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage("08123456789", null, "pesan sms", null, null);
            } catch (Exception e) {
                Log.e("SmsGatewayHandler", "Error handling message: " + e.getMessage());
            }
        }
    }
}
```

```
// ambil no tujuan dari json
String no = object.getString("no");
// ambil pesan dari json
String pesan = object.getString("pesan");

// kirim SMS
SmsManager.getDefault().sendTextMessage(no, null, pesan, null, null);

// beri respon SUKSES
httpResponse.setEntity(new StringEntity("SUKSES"));
} catch (Exception ex) {
// beri respon GAGAL
httpResponse.setEntity(new StringEntity("GAGAL"));
Log.e(SmsGatewayHandler.class.getName(), ex.getMessage(), ex);
}
}
}
```

Menambah uses-permission

Secara default, aplikasi yang kita buat itu tidak bisa mengirim SMS dan juga tidak bisa membuat ServerSocket. Jika kita memaksa, maka Android akan menolaknya, oleh karena itu kita perlu meminta permission ke user yang akan menginstall aplikasinya.

Untuk mengirim SMS kita perlu menambah permission SEND_SMS dan untuk membuat ServerSocket kita perlu menambah permission INTERNET. Untuk menambah permission, silahkan tambahkan kode dibawah ini pada file **AndroidManifest.xml**

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

Tambahkan kode diatas sebelum xml tag <application>.

Menjalankan SMS server

Sekarang kita memasuki tahapan terakhir, yaitu menjalankan SmsGatewayServer yang telah kita buat. Pertama kita tambah sebuah konstruktor dan kita override metode onDestroy() seperti pada kode dibawah ini.

```
package com.khannedy.androidsmsgateway;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    public MainActivity() {
        // TODO membuat server
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // TODO menjalankan server
    }

    @Override
    protected void onDestroy() {
        // TODO menghentikan server

        super.onDestroy();
    }
}
```

Pada konstruktor kita akan membuat objek SmsGatewayServer menggunakan port tertentu. Pada metode onCreate() kita akan jalankan server, sehingga pada saat aplikasi berjalan, otomatis servernya berjalan. Dan pada metode onDestroy() kita akan menghentikan server, sehingga saat aplikasi ditutup otomatis server akan mati. Hasilnya adalah sebagai berikut.

```
package com.khannedy.androidmsgateway;

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

import org.apache.http.HttpException;
import java.io.IOException;

public class MainActivity extends Activity {

    private SmsGatewayServer server;

    public MainActivity() {
        // membuat server
        server = new SmsGatewayServer(8989);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // menjalankan server (!!!!!CARA SALAH!!!!)
        try {
            server.start();
        } catch (IOException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        } catch (HttpException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        }
    }

    @Override
    protected void onDestroy() {
        // menghentikan server
        try {
            server.stop();
        } catch (IOException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        }

        super.onDestroy();
    }
}
```

Diatas adalah contoh kode yang salah untuk menjalankan server SmsGatewayServer nya. Kenapa salah? Ada yang tau? Krik krik krik :D

Kode diatas adalah salah karena SmsGatewayServer yang kita buat adalah server yang blocking. Dalam metode start() milik SmsGatewayServer kita melakukan perulangan while(true), alhasil saat kita panggil metode start() dari MainActivity otomatis aplikasi nya akan **HANG, bukan berarti error**, namun karena memang terjadi perulangan yang tidak berhenti untuk mendeteksi jika ada request masuk.

Oleh karena itu, karena metode start() adalah blocking, jadi untuk memanggilnya kita perlu panggil dalam Thread yang berbeda. Jadi kode yang benar untuk menjalankan server adalah sebagai berikut.

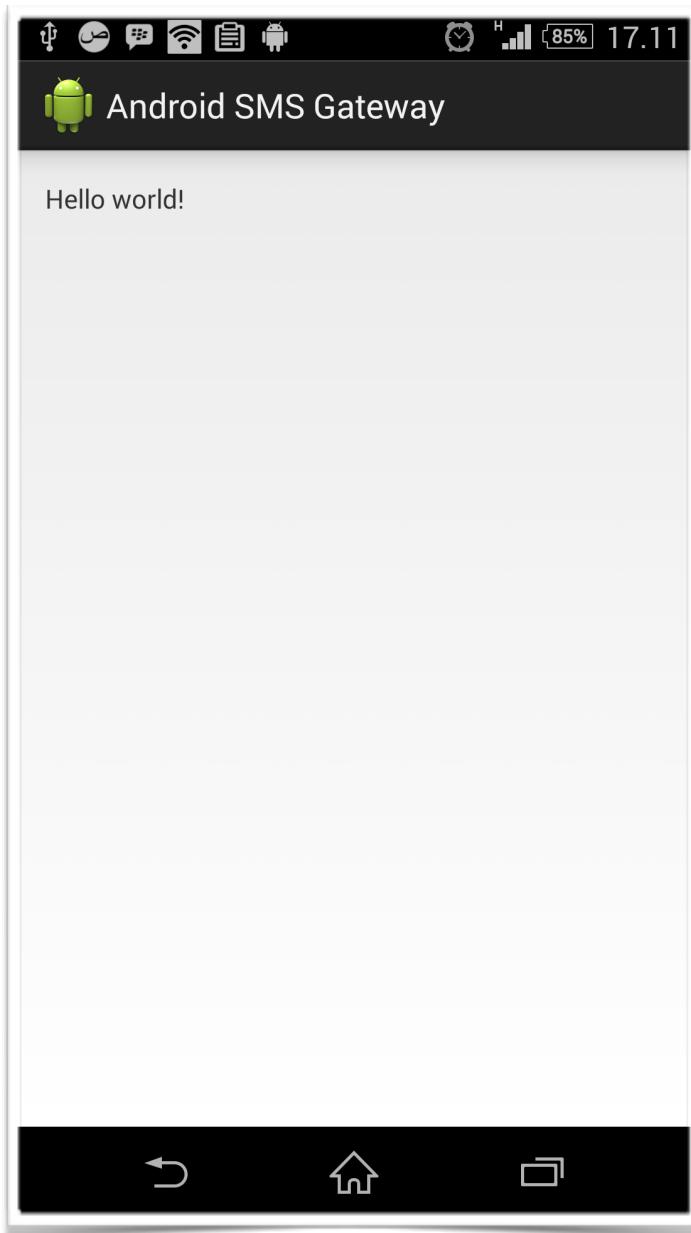
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // menjalankan server (!!!!!CARA BENAR!!!!)
    new Thread(new Runnable() {

        @Override
        public void run() {
            try {
                server.start();
            } catch (IOException e) {
                Log.e(MainActivity.class.getName(), e.getMessage(), e);
            } catch (HttpException e) {
                Log.e(MainActivity.class.getName(), e.getMessage(), e);
            }
        }
    }).start();
}
```

Dan berakhir sudah perjalanan kita membuat aplikasi Android SMS Gateway, sekarang kita coba jalankan aplikasinya. Hasilnya sama

saja dengan saat kita membuat aplikasi Blank Activity :D Karena memang dari tadi kita tidak mengubah tampilannya :D #hehehhe



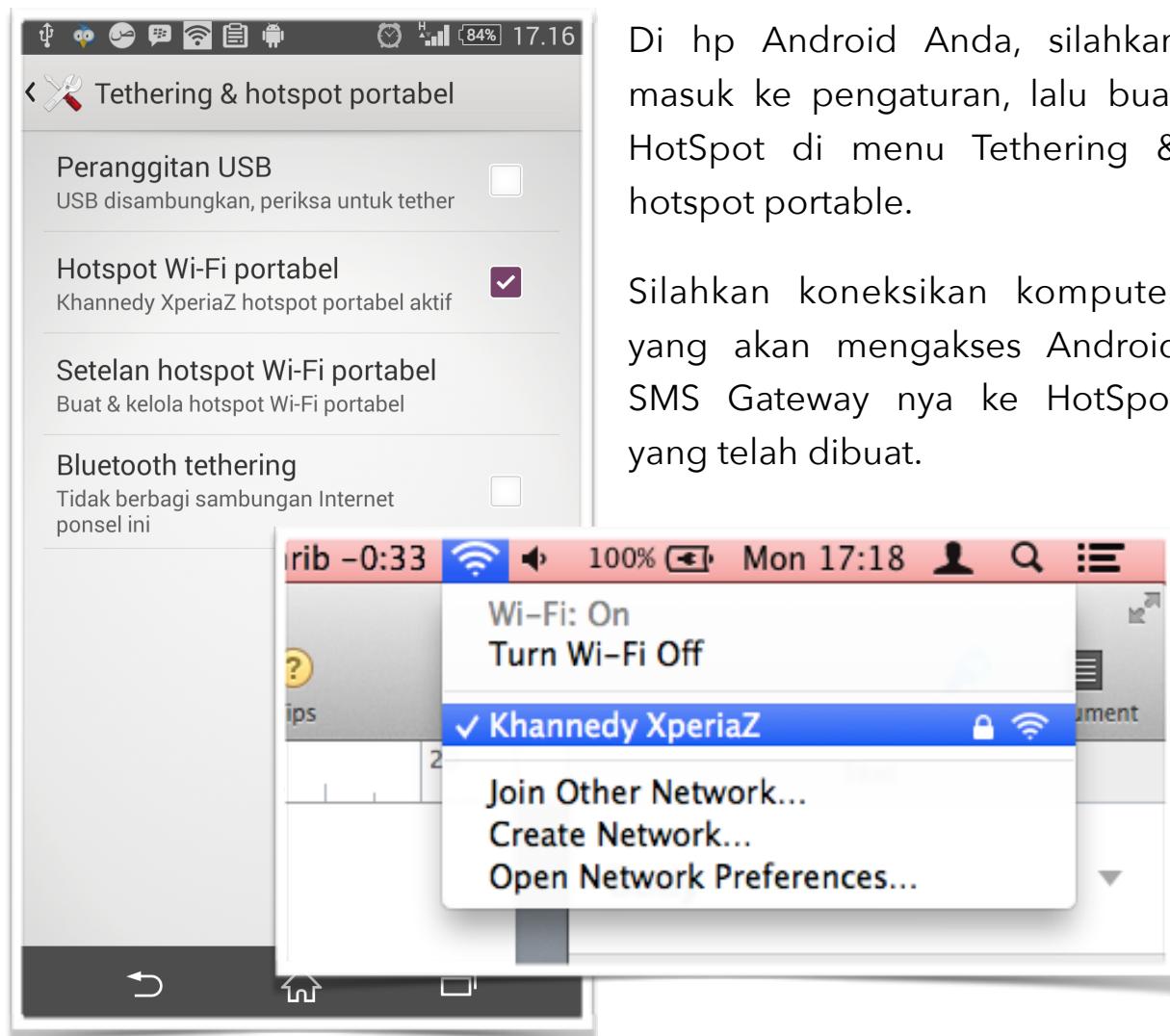
Kalo untuk UI-nya, saya serahkan ke pembaca sekalian, mau bikin gambar selfie narsis juga di UI nya terserah :D Karena gak penting UI nya kalo untuk Android SMS Gateway, yang penting fungsinya :D

Client API

ngirim sms dari aplikasi lain

Aplikasi Android SMS Gateway telah selesai, sekarang saatnya kita coba ngirim SMS dari aplikasi lain. Eh tapi sebelumnya, pastiin dulu web servernya itu bisa diakses dari jaringan wifi yang sama. Jika Anda tidak memiliki wifi, gampang saja, karena biasanya Android punya fitur untuk membuat HotSpot.

Membuat HotSpot



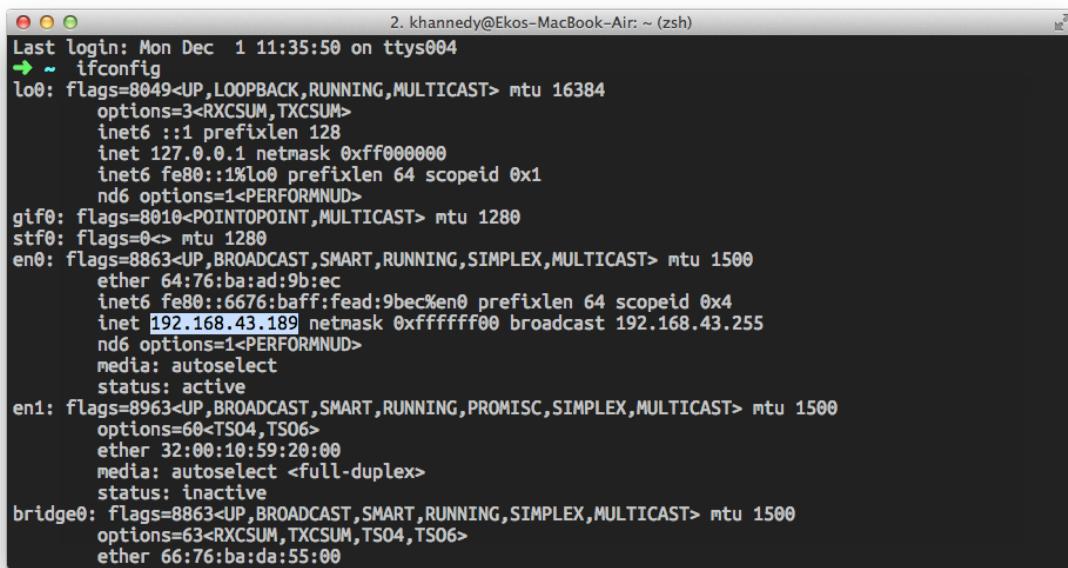
Di hp Android Anda, silahkan masuk ke pengaturan, lalu buat HotSpot di menu Tethering & hotspot portable.

Silahkan koneksi komputer yang akan mengakses Android SMS Gateway nya ke HotSpot yang telah dibuat.

Mengecek IP Address

Hal yang paling penting untuk mengirim SMS ke Android SMS Gateway adalah, kita harus tahu ip address hp Android yang menjalankan aplikasinya. Karena kita akan melakukan request HTTP ke ip address tersebut.

Untuk mengecek ip address, silahkan cek ip address komputer yang terkoneksi, jika di UNIX (Linux atau Mac) bisa menggunakan perintah **ifconfig**, kalo di Windows bisa menggunakan perintah **ipconfig**.

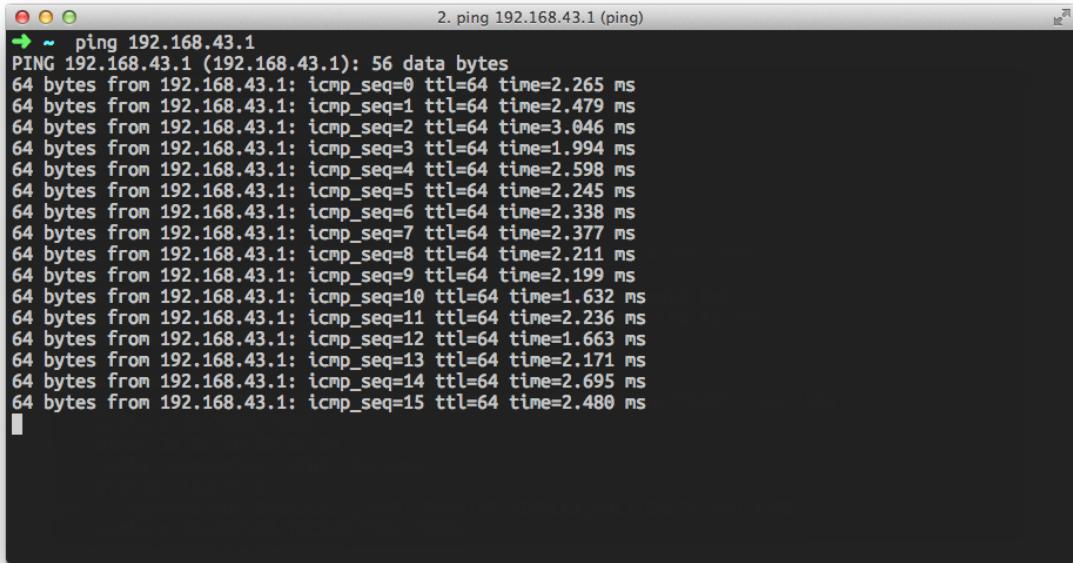


```
2. khannedy@Ekos-MacBook-Air: ~ (zsh)
Last login: Mon Dec 1 11:35:50 on ttys004
→ ~ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
            nd6 options=1<PERFORMNUD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 64:76:ba:ad:9b:ec
    inet6 fe80::6676:baff:fead:9bec%en0 prefixlen 64 scopeid 0x4
        inet 192.168.43.189 netmask 0xffffffff broadcast 192.168.43.255
            nd6 options=1<PERFORMNUD>
            media: autoselect
            status: active
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=60<TS04,TS06>
    ether 32:00:10:59:20:00
    media: autoselect <full-duplex>
    status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=63<RXCSUM,TXCSUM,TS04,TS06>
    ether 66:76:ba:da:55:00
```

Pada gambar diatas, saya mengetahui bahwa ip address komputer saya adalah **192.168.43.189**, biasanya untuk mengetahui ip address Android device-nya cukup mudah, cukup ubah nomor belakang menjadi 1 (**192.168.43.189 -> 192.168.43.1**)

Untuk memastikan apakah benar ip 192.168.43.1 adalah ip address hp Android kita, silahkan coba ping ke ip address tersebut,

jika ada response, berarti benar ip tersebut adalah ip address device Android kita.



```
2. ping 192.168.43.1 (ping)
PING 192.168.43.1 (192.168.43.1): 56 data bytes
64 bytes from 192.168.43.1: icmp_seq=0 ttl=64 time=2.265 ms
64 bytes from 192.168.43.1: icmp_seq=1 ttl=64 time=2.479 ms
64 bytes from 192.168.43.1: icmp_seq=2 ttl=64 time=3.046 ms
64 bytes from 192.168.43.1: icmp_seq=3 ttl=64 time=1.994 ms
64 bytes from 192.168.43.1: icmp_seq=4 ttl=64 time=2.598 ms
64 bytes from 192.168.43.1: icmp_seq=5 ttl=64 time=2.245 ms
64 bytes from 192.168.43.1: icmp_seq=6 ttl=64 time=2.338 ms
64 bytes from 192.168.43.1: icmp_seq=7 ttl=64 time=2.377 ms
64 bytes from 192.168.43.1: icmp_seq=8 ttl=64 time=2.211 ms
64 bytes from 192.168.43.1: icmp_seq=9 ttl=64 time=2.199 ms
64 bytes from 192.168.43.1: icmp_seq=10 ttl=64 time=1.632 ms
64 bytes from 192.168.43.1: icmp_seq=11 ttl=64 time=2.236 ms
64 bytes from 192.168.43.1: icmp_seq=12 ttl=64 time=1.663 ms
64 bytes from 192.168.43.1: icmp_seq=13 ttl=64 time=2.171 ms
64 bytes from 192.168.43.1: icmp_seq=14 ttl=64 time=2.695 ms
64 bytes from 192.168.43.1: icmp_seq=15 ttl=64 time=2.480 ms
```

Setelah kita tau ip address server yaitu **192.168.43.1**, sekarang kita harus tau port yang digunakan oleh Android SMS Gateway yang kita buat, kita liat saja di `MainActivity.java`, disana saya jalankan servernya di port **8989**.

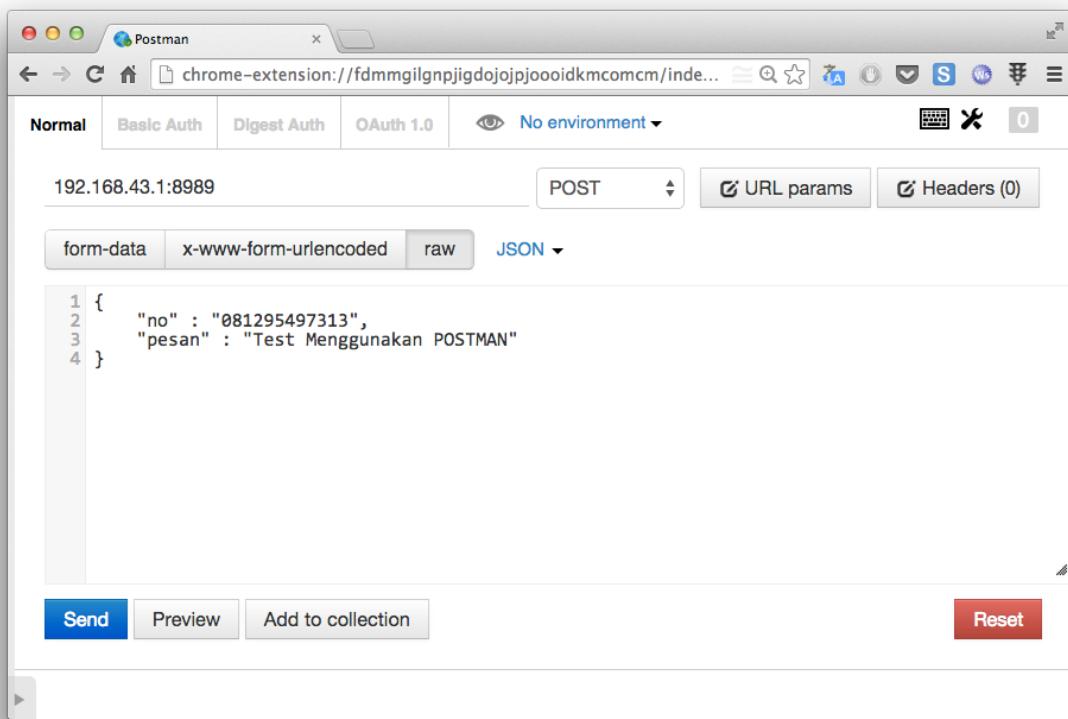
Mengirim SMS Menggunakan POSTMan

Inilah kenapa saya gunakan HTTP sebagai spesifikasi standard format pesan aplikasi Android SMS Gateway. Kita bisa menggunakan REST Client untuk mengetestnya. Salah satu REST Client yang paling saya sukai adalah POSTMan. POSTMan adalah plugin untuk Google Chrome. Anda bisa menginstallnya disini :

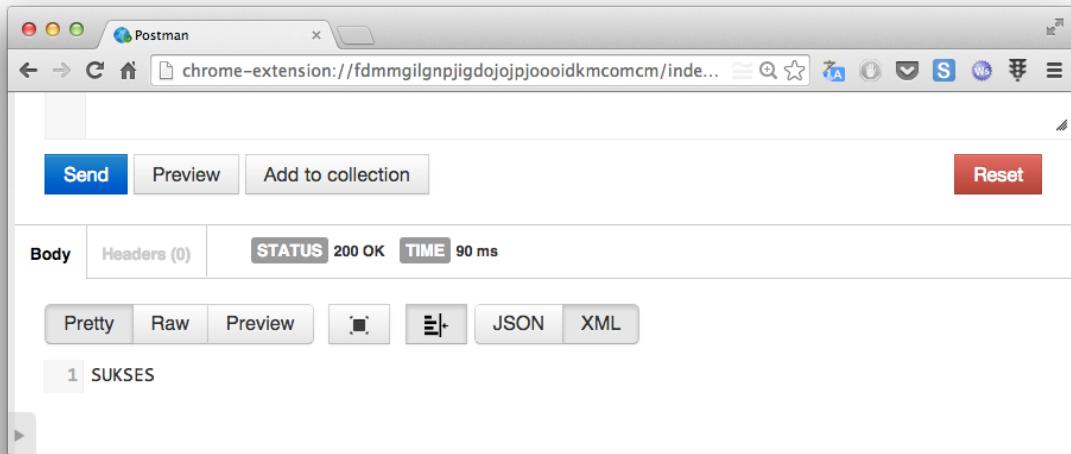
https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?utm_source=chrome-ntp-icon

Untuk mengirim SMS menggunakan POST Man cukup buat POST request ke **192.168.43.1:8989** lalu tambahkan isi pesannya berupa raw JSON, misal :

```
{  
    "no" : "081295497313",  
    "pesan" : "Test Menggunakan POSTMAN"  
}
```



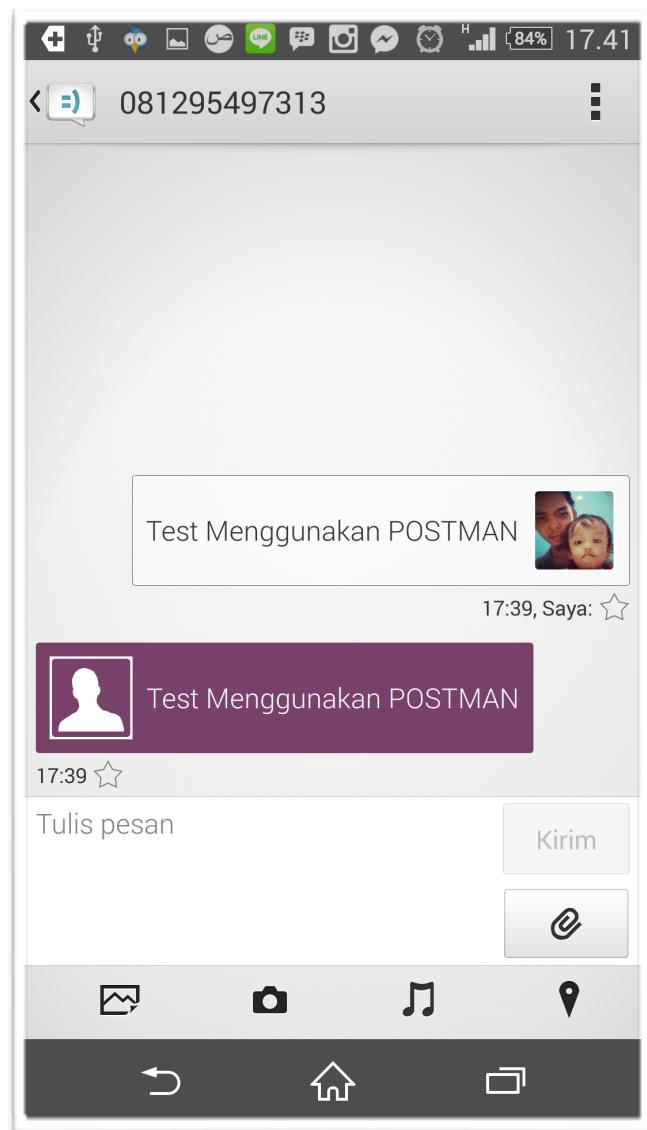
Jika kita kirim, maka POSTMan akan mengirim HTTP Request ke aplikasi Android SMS Gateway kita, dan otomatis akan mengirim pesan SMS ke nomor yang telah ditentukan. Hasilnya pastikan bahwa respon dari Android SMS Gateway kita adalah SUKSES seperti pada gambar dibawah ini.



Dan hasilnya seperti pada INBOX SMS saya, Saya mengirim pesan dari nomor yang sama, jadi terlihat pesannya 2 kali, mengirim dan menerima SMS.

Dengan ini berarti kita telah berhasil membuat aplikasi Android SMS Gateway. Saya ucapkan SELAMAT!!!!

Selanjutnya akan saya beri contoh beberapa kode untuk mengirim sms via Android SMS Gateway menggunakan beberapa bahasa pemrograman.



Mengirim SMS Menggunakan Java

Berikut adalah kode Java (menggunakan Apache HTTP Client dan Google GSON) untuk mengirim SMS via Android SMS Gateway.

```
package com.khannedy;

import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;

import java.io.IOException;

/**
 * @author Eko Khannedy
 * @since 12/1/14
 */
public class SendSMSApp {

    public static void main(String[] args) throws IOException {
        // membuat json
        JsonObject object = new JsonObject();
        object.add("no", new JsonPrimitive("081295497313"));
        object.add("pesan", new JsonPrimitive("Pesan dari JAVA"));

        // konversi ke String JSON
        Gson gson = new Gson();
        String json = gson.toJson(object);

        // mengirim json ke Android SMS Server
        HttpClient httpClient = HttpClients.createDefault();
        HttpPost post = new HttpPost("http://192.168.43.1:8989");
        post.setEntity(new StringEntity(json));
        httpClient.execute(post);
    }
}
```

Mengirim SMS dari NodeJS

Berikut adalah contoh kode mengirim SMS menggunakan NodeJS via Android SMS Gateway.

```
var http = require("http");

var options = {
  hostname: '192.168.43.1',
  port: 8989,
  path: '/',
  method: 'POST'
};

var request = {
  "no": "081295497313",
  "pesan": "Pesanan dari NodeJS"
};

var json = JSON.stringify(request);

var req = http.request(options, function (res) {
  res.on('data', function (chunk) {
    console.log('Response : ' + chunk);
  });
});

req.on('error', function (e) {
  console.log('problem with request: ' + e.message);
});

req.write(json);
req.end();
```

Mengirim SMS Menggunakan PHP

Berikut adalah kode yang bisa kita gunakan untuk mengirim SMS menggunakan PHP via Android SMS Gateway.

```
$data = array("no" => "081295497313", "pesan" => "SMS dari PHP");
$data_string = json_encode($data);

$ch = curl_init('http://192.168.43.1:8989');
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_POSTFIELDS, $data_string);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-Length: ' . strlen($data_string))
);

$result = curl_exec($ch);
```

Mengirim SMS Menggunakan Ruby

Berikut adalah kode yang bisa digunakan untuk mengirim SMS menggunakan Ruby (menggunakan HTTParty gems) via Android SMS Gateway.

```
require 'httparty'

HTTParty.post("http://192.168.43.1:8989",
  body: {
    no : "081295497313",
    pesan : "SMS dari Ruby"
  }
).body
```

Mungkin cukup itu saja, jika Anda menggunakan teknologi atau bahasa pemrograman yang lain, tinggal cari saja implementasi HTTP Client di masing-masing teknologi atau bahasa pemrograman yang Anda gunakan :)

What's Next?

yang perlu dilakukan selanjutnya

Aplikasi Android yang kita buat tadi belum benar-benar sempurna, berikut adalah beberapa tugas yang perlu anda lakukan agar program Android SMS Gateway yang telah kita buat menjadi sempurna.

- ▶ Validasi format JSON, tidak ada validasi di server jika user mengirim format JSON yang salah, silahkan lakukan validasi di Server SMS Gateway nya
- ▶ Validasi no telepon, tidak ada validasi di server jika user mengirim data no telepon yang salah, misal "no salah", silahkan lakukan validasi di Server SMS Gateway
- ▶ Buat port portable, saat ini aplikasi server berjalan dengan port yang telah ditentukan, silahkan buat agar port server bisa ditentukan saat aplikasi running, misal buat tombol START, tombol STOP lalu input PORT, jadi saat user mengklik tombol START, server akan berjalan sesuai port yang ada di input PORT. Dan saat tombol STOP di klik, server otomatis berhenti.

Selamat berjuang! Terimakasih telah membaca buku FREE EDITION, semoga ilmunya bermanfaat :D

Rombak Ulang App

kita akan buat dari awal lagi #yeah

Di PREMIUM EDITION, kita sama sekali tidak akan melanjutkan aplikasi yang terdapat di FREE EDITION, justru disini kita akan merombak ulang, atau lebih telahnya re-write kembali project Android SMS Gateway dari awal.

Agar lebih cepat, dari pada kita ubah aplikasi Android SMS Gateway sebelumnya yang sudah kita buat. Lebih baik sekarang kita buat project baru saja. Silahkan beri nama project tersebut dengan nama **Android SMS Gateway PREMIUM**.

Kenapa HTTP Request Tidak Cocok?

Sebelumnya kita telah membuat sebuah aplikasi Android SMS Gateway menggunakan HTTP Server. Aplikasi tersebut memang sudah bagus, namun banyak kendala jika kita menggunakan HTTP Server. Kita tidak akan bisa melakukan request dari server (Android SMS Gateway) ke client (Application). Karena yang namanya HTTP request itu dilakukan dari client ke server, dan tidak sebaliknya.

Keterbatasan ini menjadikan kita cukup sulit untuk mengimplementasikan fitur "Menerima SMS Secara REAL-TIME", karena jika ingin REAL-TIME, kita harus bisa melakukan hal sebaliknya, yaitu mengirim request dari server ke client. Maka dari itu, di bagian PREMIUM EDITION ini, kita akan buat dari awal menggunakan teknologi yang berbeda.

Teknologi apa yang cocok agar bisa komunikasi dua arah? Dan juga teknologinya mudah diimplementasikan? Dan mudah diintegrasikan?

ServerSocket manual? Ohhhh tidakkkkk!!! Jangan manual implementasi ServerSocket, selain kita harus maintain Thread per Request secara manual, itu juga mempersulit kita dalam melakukan integrasi, karena SocketClient juga harus dibuat secara manual di tiap client-nya.

So teknologi apa yang cocok? Jawabannya adalah teknologi **WebSocket!** Teknologi yang dikenalkan dalam HTML5. Dengan WebSocket kita bisa melakukan komunikasi dua arah antara client dan server, dan kabar yang paling baiknya, hampir semua teknologi sudah mengimplementasikan teknologi WebSocket.

Memilih Framework WebSocket

Mengimplementasikan WebSocket secara manual bukanlah hal yang bijak :D Karena tujuan kita bukanlah membuat framework WebSocket, namun tujuan kita untuk menggunakan WebSocket hanyalah untuk mempermudah membuat Android SMS Gateway PREMIUM EDITION #hehehehe

Salah satu framework yang bagus untuk implementasi WebSocket Server adalah **Java-WebSocket**. Walaupun untuk Java, bagusnya framework Java-WebSocket bisa jalan di platform Android. Dan yang paling keren, framework ini sangat ringan dan mudah sekali digunakan :D #yay

Ini adalah website resmi dari framework Java-WebSocket : <http://java-websocket.org/>. Anda bisa jalan-jalan terlebih dahulu di websitenya sebelum melanjutkan membaca buku ini :D

Menambah Library Java-WebSocket

Untuk menggunakan Java-WebSocket, tambahkan terlebih dahulu library Java-WebSocket nya. Android Studio menggunakan Gradle untuk build tool nya, jadi kita cukup tambahkan library nya di **build.gradle** nya.

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
  
    // Java-WebSocket Library  
    compile 'org.java-websocket:Java-WebSocket:1.3.0'  
}
```

Silahkan **Sync** ulang projectnya di Android Studio agar Android Studio secara otomatis men-download library Java-WebSocket nya.

Jika Anda menggunakan ADT, maka Anda perlu mendownload secara manual library Java-WebSocket nya di website resminya. Maka dari itu saya lebih suka menggunakan Android Studio dibandingkan menggunakan ADT.

Membuat SmsGatewayContainer

Sebelum kita mulai membuat WebSocket Server untuk SMS Gateway-nya, terlebih dahulu kita buat SmsGatewayContainer untuk menyimpan data client yang terkoneksi ke Server. Kita akan buat class nya dalam bentuk utilities class, sehingga semua metodenya kita buat static.

Kita akan buat 2 metode, yaitu add() untuk menambah WebSocket client yang terkoneksi, dan remove() untuk menghapus WebSocket client yang diskonek dari server.

```
package com.khannedy.androidsmsgatewaypremium;

import org.java_websocket.WebSocket;

import java.util.ArrayList;
import java.util.List;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayContainer {

    /**
     * org.java_websocket.WebSocket adalah implementasi WebSocket client
     * di Java-WebSocket
     */
    private static List<WebSocket> sockets = new ArrayList<WebSocket>();

    /**
     * Menambah WebSocket client ke container
     *
     * @param socket WebSocket client
     */
    public static void add(WebSocket socket) {
        sockets.add(socket);
    }

    /**
     * Menghapus WebSocket client dari container
     *
     * @param socket WebSocket client
     */
    public static void remove(WebSocket socket) {
        sockets.remove(socket);
    }
}
```

Membuat SmsGatewayServer

Untuk membuat WebSocket menggunakan Java-WebSocket, kita hanya perlu membuat kelas turunan dari `WebSocketServer`. Lalu

implementasi beberapa metode abstractnya. Sekarang mari kita buat kelas SmsGatewayServer yang memiliki parent class WebSocketServer.

```
package com.khannedy.androidsmsgatewaypremium;

import org.java_websocket.WebSocket;
import org.java_websocket.handshake.ClientHandshake;
import org.java_websocket.server.WebSocketServer;

import java.net.InetSocketAddress;
import java.net.UnknownHostException;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayServer extends WebSocketServer {

    public SmsGatewayServer(int port) throws UnknownHostException {
        super(new InetSocketAddress(port));
    }

    @Override
    public void onOpen(WebSocket conn, ClientHandshake handshake) {
        // tambahkan client ke container
        SmsGatewayContainer.add(conn);
    }

    @Override
    public void onClose(WebSocket conn, int code, String reason, boolean remote) {
        // hapus client dari container
        SmsGatewayContainer.remove(conn);
    }

    @Override
    public void onMessage(WebSocket conn, String message) {
    }

    @Override
    public void onError(WebSocket conn, Exception ex) {
    }
}
```

Ada beberapa metode yang abstract sehingga kita perlu mengimplementasikannya di kelas SmsGatewayServer.

- ▶ **onOpen()** dipanggil ketika ada koneksi WebSocket client baru, dan setiap ada client baru, kita tambahkan ke SmsGatewayContainer.
- ▶ **onClose()** dipanggil ketika ada koneksi yang terputus dari server, dan setiap ada koneksi yang terputus, kita hapus dari SmsGatewayContainer.
- ▶ **onMessage()** dipanggil ketika WebSocket client mengirim pesan ke server
- ▶ **onError()** dipanggil ketika terjadi error dengan koneksi WebSocket client.

Saat ini kita hanya implementasikan isi metode **onOpen()** dan **onClose()** dulu, nanti kita implementasikan metode **onMessage()** nya.

Format Pesan Request

Format pesan yang akan kita gunakan adalah JSON, yup JSON sama saja seperti aplikasi di FREE EDITION, bedanya mungkin nama atributnya kita ganti menjadi bahasa inggris #gayaDikit :D

```
{  
    "to" : "081295497313",  
    "message" : "Hello, how are you?"  
}
```

Mengirim SMS

Setelah format pesan JSON nya jelas, sekarang kita akan kirim SMS dari pesan yang telah diterima. Caranya tinggal kita implementasikan saja metode **onMessage()** di SmsGatewayServer-nya.

```
@Override
public void onMessage(WebSocket conn, String message) {
    try {
        JSONObject object = new JSONObject(message);

        String to = object.getString("to");
        String smsMessage = object.getString("message");

        SmsManager.getDefault().sendTextMessage(to, null, smsMessage, null, null);
    } catch (JSONException e) {
        conn.send("Format JSON salah");
    }
}
```

Sekarang kita telah selesai mengimplementasikan jika ada WebSocket client yang ingin mengirim SMS. Saatnya kita jalankan Server nya saat aplikasi Android nya berjalan.

Menjalankan SMS Gateway Server

Untuk menjalankan SmsGatewayServer caranya hampir sama dengan di FREE EDITION, WebSocketServer memiliki metode start() untuk menjalankan server dan metode stop() untuk menghentikan server. Perbedaannya adalah, metode start() milik WebSocketServer tidak blocking, jadi kita tidak perlu memanggilnya dalam new Thread #yay

```
package com.khannedy.androidmsgatewaypremium;

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

import java.io.IOException;
import java.net.UnknownHostException;

public class MainActivity extends Activity {

    private SmsGatewayServer server;

    public MainActivity() {
        try {
            // membuat server dengan port 8989
            server = new SmsGatewayServer(8989);
        } catch (UnknownHostException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // menjalankan server
        server.start();
    }

    @Override
    protected void onDestroy() {

        // menghentikan server
        try {
            server.stop();
        } catch (IOException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        } catch (InterruptedException e) {
            Log.e(MainActivity.class.getName(), e.getMessage(), e);
        }
    }

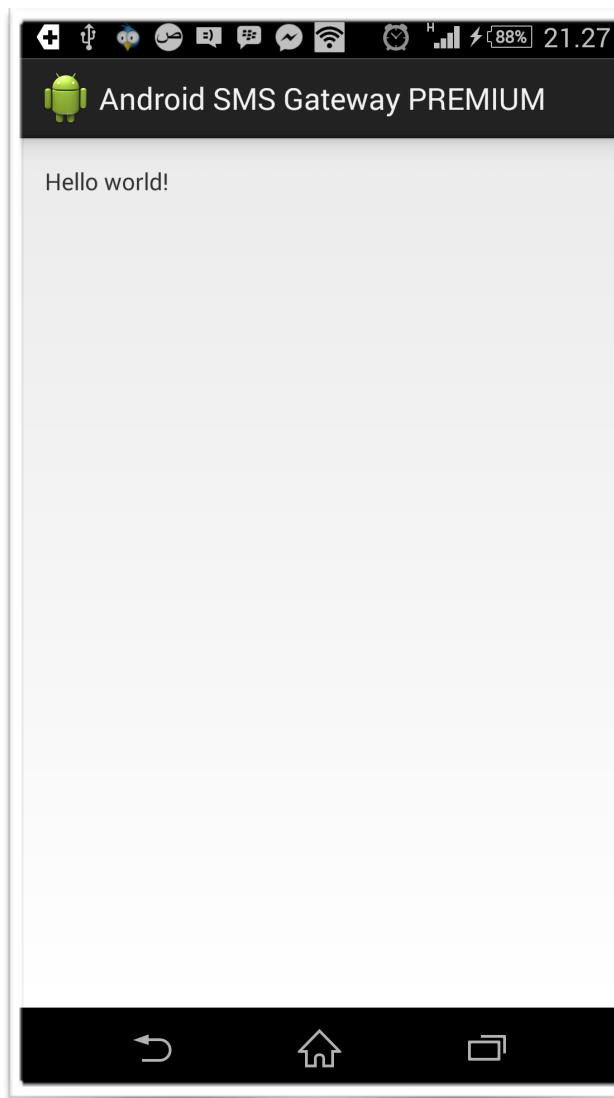
    super.onDestroy();
}
}
```

Menambah Permission

Jangan lupa untuk menambahkan uses-permission di AndroidManifest.xml. Untuk uses-permission nya sama saja dengan di FREE EDITION, yaitu **SEND_SMS** dan **INTERNET**.

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

Terakhir silahkan jalankan aplikasinya :D



Client API

untuk PREMIUM EDITION

Di FREE EDITION, kita menggunakan library HTTP Client untuk tiap teknologi. Namun untuk sekarang saya hanya akan bahas satu teknologi saja, yaitu Java. Untuk teknologi yang lainnya Anda tinggal cari tau saja cara membuat WebSocket Client di NodeJS, PHP dan Ruby misalnya.

Menambah Library Java-WebSocket

Client Java yang akan saya buat akan menggunakan Apache Maven sebagai build tool-nya. Jika Anda telah membuat project menggunakan Apache Maven (Hampir Semua IDE (Eclipse, NetBeans, IntelliJ IDEA) mendukung Apache Maven), silahkan tambahkan library Java-WebSocket di pom.xml nya

```
<dependencies>
    <!--Dependency Java-WebSocket-->
    <dependency>
        <groupId>org.java-websocket</groupId>
        <artifactId>Java-WebSocket</artifactId>
        <version>1.3.0</version>
    </dependency>
</dependencies>
```

Membuat WebSocketClient

Hampir sama saat membuat Android SMS Gateway Server, kita juga perlu membuat hal yang sama, bedanya kita sekarang membuat WebSocketClient. Misal Kita buat dengan nama SmsGatewayClient.

```
package com.khannedy.premium;

import org.java_websocket.client.WebSocketClient;
import org.java_websocket.handshake.ServerHandshake;

import java.net.URI;

/**
 * @author Eko Khannedy
 * @since 12/1/14
 */
public class SmsGatewayClient extends WebSocketClient {

    public SmsGatewayClient(URI serverURI) {
        super(serverURI);
    }

    @Override
    public void onOpen(ServerHandshake serverHandshake) {
        System.out.println("KONEKSI TERSAMBUNG");
    }

    @Override
    public void onMessage(String s) {
        System.out.println("SERVER : " + s);
    }

    @Override
    public void onClose(int i, String s, boolean b) {
        System.out.println("KONEKSI TERPUTUS");
    }

    @Override
    public void onError(Exception e) {
        System.out.println("TERJADI ERROR");
    }
}
```

Client yang kita buat hanya sederhana, yaitu menampilkan tulisan di console.

Menambah Library Google GSON

Agar lebih mudah membuat pesan JSON nya, saya sarankan menggunakan Google GSON, jadi silahkan tambahkan Google GSON ke dependency Apache Maven nya di pom.xml

```
<dependencies>
    <!--Dependency Java-WebSocket-->
    <dependency>
        <groupId>org.java-websocket</groupId>
        <artifactId>Java-WebSocket</artifactId>
        <version>1.3.0</version>
    </dependency>
    <!--Dependency Google Gson-->
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.3</version>
    </dependency>
</dependencies>
```

Mengirim Pesan ke Server

Sekarang kita jalankan SmsGatewayClient agar dapat mengirim pesan ke server SmsGatewayServer. Berikut adalah contohnya.

```
package com.khannedy.premium;

import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

import java.net.URI;
import java.net.URISyntaxException;

/**
 * @author Eko Khannedy
 * @since 12/1/14
 */
public class App {

    public static void main(String[] args)
        throws URISyntaxException, InterruptedException {

        URI server = new URI("ws://192.168.43.1:8989");
        SmsGatewayClient client = new SmsGatewayClient(server);

        // menyalakan koneksi
        client.connectBlocking();

        // membuat Json message
        JsonObject object = new JsonObject();
        object.add("to", new JsonPrimitive("081295497313"));
        object.add("message", new JsonPrimitive("Hello PREMIUM"));

        // mengonversi ke JSON String
        Gson gson = new Gson();
        String json = gson.toJson(object);

        // mengirim pesan ke server
        client.send(json);

        // memutuskan koneksi
        client.closeBlocking();
    }
}
```

Selamat! Kita telah berhasil membuat Client yang menggunakan Android SMS Gateway **PREMIUM** :D Sekarang kita lanjutkan dengan fitur yang baru di **PREMIUM EDITION**.

Menerima Laporan

sukses atau gagal pesan SMS nya

Sampai saat ini kita belum terlalu memaksimalkan penggunaan SmsGatewayContainer. Pada fitur ini, kita benar-benar akan memanfaatkan SmsGatewayContainer.

Salah satu kekurangan pada aplikasi Android SMS Gateway PREMIUM yang telah kita buat sebelumnya adalah, kita tidak tau status dari pesan SMS yang dikirim, apakah sukses atau gagal (misal karena pulsa abis :P). Jadi sekarang kita akan buat fitur agar server bisa mengirimkan status terakhir dari SMS yang telah dikirimkan.

Membuat metode send() di SmsGatewayContainer

Pertama yang akan kita lakukan adalah, kita akan membuat sebuah metode send() di **SmsGatewayContainer** untuk melakukan pengiriman pesan ke client yang terkoneksi. Metodenya cukup mudah implementasinya.

```
/**  
 * Mengirim pesan dari server ke client  
 *  
 * @param message pesan  
 */  
public static void send(String message) {  
    for (WebSocket socket : sockets) {  
        socket.send(message);  
    }  
}
```

Menambahkan PendingIntent

Di metode sendTextMessage di SmsManager sebenarnya kita bisa menambahkan PendingIntent untuk mengetahui status dari SMS yang kita kirim, apakah sukses apakah gagal beserta alasan kenapa gagalnya. Oleh karena itu, sekarang kita akan buat PendingIntent untuk mengetahui status pengiriman SMS.

```
SmsManager.getDefault().sendTextMessage(to, null, message, sendIntent, deliveryIntent);
```

Silahkan tambahkan dua buat atribut PendingIntent di SmsGatewayServer, dan tambahkan metode setter agar bisa diubah dari luar seperti pada kode dibawah ini.

```
import android.app.PendingIntent;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayServer extends WebSocketServer {

    private PendingIntent sendIntent;

    private PendingIntent deliveryIntent;

    public SmsGatewayServer(int port) throws UnknownHostException {
        super(new InetSocketAddress(port));
    }

    public void setSendIntent(PendingIntent sendIntent) {
        this.sendIntent = sendIntent;
    }

    public void setDeliveryIntent(PendingIntent deliveryIntent) {
        this.deliveryIntent = deliveryIntent;
    }
}
```

Selanjutnya jangan lupa tambahkan atribut **sendIntent** dan **deliveryIntent** ke metode sendTextMessage SmsManager di metode onMessage()

```
@Override
public void onMessage(WebSocket conn, String message) {
    try {
        JSONObject object = new JSONObject(message);

        String to = object.getString("to");
        String smsMessage = object.getString("message");

        SmsManager.getDefault()
            .sendTextMessage(to, null, smsMessage, sendIntent, deliveryIntent);

    } catch (JSONException e) {
        conn.send("Format JSON salah");
    }
}
```

Menambahkan PendingIntent dari MainActivity

PendingIntent yang telah kita buat (sendIntent dan deliveryIntent) perlu di supply dari luar kelas SmsGatewayServer. Kita akan mensupply objek nya dari MainActivity sebelum memanggil metode start() server.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // buat sendIntent
    PendingIntent sendIntent = PendingIntent.getBroadcast(this, 0,
        new Intent("SMS_SENT"), 0);

    // buat deliveryIntent
    PendingIntent deliveryIntent = PendingIntent.getBroadcast(this, 0,
        new Intent("SMS_DELIVERED"), 0);

    // set sendIntent dan deliveryIntent
    server.setSendIntent(sendIntent);
    server.setDeliveryIntent(deliveryIntent);

    // menjalankan server
    server.start();
}
```

Sekarang sebelum server berjalan, kita telah men-supply nilai sendIntent dan pendingIntent nya. Namun tetap belum bisa memberi informasi ke client tentang status SMS yang dikirim nya.

Membuat Receiver

Agar status terkini SMS yang dikirim oleh PendingIntent dapat diterima, kita perlu menambahkan receiver untuk tiap PendingIntent nya, yaitu receiver untuk sendIntent dan receiver untuk deliveryIntent. Caranya adalah tambahkan menggunakan metode **registerReceiver()** di metode **onCreate()** kelas **MainActivity** sebelum server berjalan.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // receiver untuk sendIntent
    registerReceiver(new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            switch (getResultCode()) {
                case Activity.RESULT_OK:
                    SmsGatewayContainer.send("SMS sent");
                    break;
                case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                    SmsGatewayContainer.send("Generic failure");
                    break;
                case SmsManager.RESULT_ERROR_NO_SERVICE:
                    SmsGatewayContainer.send("No service");
                    break;
                case SmsManager.RESULT_ERROR_NULL_PDU:
                    SmsGatewayContainer.send("Null PDU");
                    break;
                case SmsManager.RESULT_ERROR_RADIO_OFF:
                    SmsGatewayContainer.send("Radio off");
                    break;
            }
        }
    }, new IntentFilter("SMS_SENT"));

    // receiver untuk deliveryIntent
    registerReceiver(new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            switch (getResultCode()) {
                case Activity.RESULT_OK:
                    SmsGatewayContainer.send("SMS delivered");
                    break;
                case Activity.RESULT_CANCELED:
                    SmsGatewayContainer.send("SMS not delivered");
                    break;
            }
        }
    }, new IntentFilter("SMS_DELIVERED"));
}
```

```
// buat sendIntent
PendingIntent sendIntent = PendingIntent.getBroadcast(this, 0,
    new Intent("SMS_SENT"), 0);

// buat deliveryIntent
PendingIntent deliveryIntent = PendingIntent.getBroadcast(this, 0,
    new Intent("SMS_DELIVERED"), 0);

// set sendIntent dan deliveryIntent
server.setSendIntent(sendIntent);
server.setDeliveryIntent(deliveryIntent);

// menjalankan server
server.start();
}
```

Sekarang setiap ada perubahan status pengiriman SMS, maka client akan diberi tahu, bahkan jika ada error pun client akan diberi tahu.

Sekarang, kita telah mengimplementasikan laporan status pengiriman SMS di PREMIUM EDITION, selanjutnya kita akan implementasikan fitur untuk menerima SMS secara REAL-TIME.

Menerima SMS

secara REAL-TIME saat itu juga

Fitur yang paling penting dalam SMS Gateway adalah menerima SMS. Kurang keren kalo aplikasi SMS Gateway hanya bisa mengirim SMS tapi tidak bisa menerima SMS :D

Fitur ini sebenarnya sangat mudah untuk diimplementasikan :D #hehehehe. Tidak percaya? Oke ayo kita implementasikan!

Membuat BroadcastReceiver

Agar SMS yang masuk dapat otomatis diterima oleh aplikasi Android SMS Gateway PREMIUM yang kita buat, kita perlu membuat sebuah class turunan BroadcastReceiever, dimana kelas ini akan dipanggil oleh Android ketika ada SMS masuk. Misal kita buat kelas dengan nama **SmsGatewayReceiver**.

```
package com.khannedy.androidsmsgatewaypremium;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

    }
}
```

Tinggal kita implementasikan receiver untuk menerima SMS di metode onReceive() seperti pada kode dibawah ini.

```
package com.khannedy.androidmsgatewaypremium;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;

/**
 * @author Eko Khannedy
 */
public class SmsGatewayReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();

        if (bundle != null) {
            // menerima pesan SMS
            Object[] pdus = (Object[]) bundle.get("pdus");
            for (Object pdu : pdus) {
                // conversi ke SmsMessage
                SmsMessage message = SmsMessage.createFromPdu((byte[]) pdu);
                String string = "SMS from " + message.getOriginatingAddress() +
                    " : " + message.getMessageBody();

                // kirim ke client
                SmsGatewayContainer.send(string);
            }
        }
    }
}
```

Secara default, BroadcastReceiver yang telah kita buat tidak akan diakses otomatis oleh Android ketika ada SMS masuk. Kita perlu meregistrasikannya ke AndroidManifest.xml terlebih dahulu.

Meregistrasikan BroadcastReceiver

Untuk meregistrasikan BroadcastReceiver kita perlu menambahkan <receiver> di AndroidManifest.xml. Tambahkan tag <receiver> dalam tag <application> seperti pada kode dibawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.khannedy.androidsmsgatewaypremium">

    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">

        <activity
            android:name="com.khannedy.androidsmsgatewaypremium.MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!--Menambahkan BroadcastReceiver-->
        <receiver android:name="com.khannedy.androidsmsgatewaypremium.SmsGatewayReceiver">
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Menamba uses-permission RECEIVE_SMS

Terakhir sebelum bisa mencoba fitur menerima SMS secara REAL-TIME, kita harus menambahkan uses-permission untuk

RECEIVE_SMS di AndroidManifest.xml. Silahkan tambahkan dulu uses-permission nya seperti pada kode dibawah ini.

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

Selesai!

Sekaran Anda bisa menjalankan Aplikasi Android SMS Gateway PREMIUM nya, dan coba jalankan kembali aplikasi clientnya. Sekarang setiap Ada SMS client akan mendapat notifikasi secara REAL-TIME, misal seperti berikut :

SMS from 3636 : Pemakaian kuota internet siang anda akan dialihkan ke kuota malam pkl 00.00. Cek sisa kuota Anda secara berkala di *363#. Tariff normal berlaku jk quota tlh habis

#hehehehhe

Revisi Dikit

revisi dikit biar lebih ZUPER KEREN!

Kita akan melakukan revisi sedikit dari aplikasi Android SMS Gateway PREMIUM yang telah kita buat. Apa itu? Namanya kita ganti jadi ULTIMATE #eh bukan-bukan :D

Pada aplikasi PREMIUM, terdapat hal yang tidak konsisten. Apa itu? **Pesan respon dari server!** Client mengirim pesan berupa JSON, sedangkan Server mengirim pesan dengan format Text biasa. Hal ini akan sulit dimengerti oleh Client, jenis pesan apa yang dikirim oleh Server. Apakah jenis pesan notifikasi status pengiriman SMS atau ada pesan SMS baru yang masuk.

Oleh karena itu, kita akan ubah format pesannya menjadi JSON untuk response dari Server ke Client. Berikut adalah format standard nya :

Ketika client mengirim format JSON yang salah, maka responsenya adalah sebagai berikut :

```
{  
  "type": "error",  
  "message": "Wrong JSON format"  
}
```

Ketika client mengirim format JSON yang benar, dan server melakukan pengiriman SMS, responsenya adalah sebagai berikut :

```
{  
  "type": "success",  
  "message": "Success send SMS"  
}
```

Ketika client menerima notifikasi status pengiriman pesan, maka responsenya adalah sebagai berikut :

```
{  
  "type": "notification",  
  "message": "SMS Delivered",  
  "success": true  
}
```

```
{  
  "type": "notification",  
  "message": "No Service",  
  "success": false  
}
```

Ketika client menerima notifikasi bahwa ada SMS yang masuk, maka responsenya adalah sebagai berikut :

```
{  
  "type": "received",  
  "from": "089111111111",  
  "message": "Hello, apa kabar bro?"  
}
```

Dengan begitu sekarang untuk mengetahui jenis pesan yang diterima, kita tinggal cek "**type**" nya apa.

Revisi onMessage() SmsGatewayServer

Untuk mengimplementasikan standarisasi JSON, pertama kita revisi metode onMessage() milik kelas SmsGatewayServer menjadi seperti berikut :

```
@Override
public void onMessage(WebSocket conn, String message) {
    try {
        JSONObject object = new JSONObject(message);

        String to = object.getString("to");
        String smsMessage = object.getString("message");

        SmsManager.getDefault().
            sendTextMessage(to, null, smsMessage, sendIntent, deliveryIntent);

        // kirim pesan sukses
        Map<String, String> map = new HashMap<String, String>();
        map.put("type", "success");
        map.put("message", "Success Send SMS");
        JSONObject response = new JSONObject(map);
        conn.send(response.toString());
    } catch (JSONException e) {
        // kirim pesan error
        Map<String, String> map = new HashMap<String, String>();
        map.put("type", "error");
        map.put("message", "Wrong JSON format");
        JSONObject response = new JSONObject(map);
        conn.send(response.toString());
    }
}
```

Revisi Laporan Pengiriman SMS

Khusus untuk laporan pengiriman, kita akan tambahkan sebuah metode baru di SmsGatewayContainer, bernama notification(), dimana terdapat parameter message:String dan success:Boolean, disini untuk mengetahui apakah status pengiriman SMS itu berhasil atau gagal.

```
/**  
 * Mengirim notifikasi status pengiriman SMS  
 *  
 * @param message pesan  
 * @param success sukses atau gagal  
 */  
public static void notification(String message, boolean success) {  
    Map<String, Object> map = new HashMap<String, Object>();  
    map.put("type", "notification");  
    map.put("message", message);  
    map.put("success", success);  
    JSONObject response = new JSONObject(map);  
  
    // panggil metode send  
    send(response.toString());  
}
```

Setelah dibuat metode notification, tinggal ubah cara mengirim notifikasi dari receivernya yang terdapat di MainActivity.java

```
// receiver untuk sendIntent  
registerReceiver(new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        switch (getResultCode()) {  
            case Activity.RESULT_OK:  
                SmsGatewayContainer.notification("SMS sent", true);  
                break;  
            case SmsManager.RESULT_ERROR_GENERIC_FAILURE:  
                SmsGatewayContainer.notification("Generic failure", false);  
                break;  
            case SmsManager.RESULT_ERROR_NO_SERVICE:  
                SmsGatewayContainer.notification("No service", false);  
                break;  
            case SmsManager.RESULT_ERROR_NULL_PDU:  
                SmsGatewayContainer.notification("Null PDU", false);  
                break;  
            case SmsManager.RESULT_ERROR_RADIO_OFF:  
                SmsGatewayContainer.notification("Radio off", false);  
                break;  
        }  
    }  
}, new IntentFilter("SMS_SENT"));
```

```
// receiver untuk deliveryIntent
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                SmsGatewayContainer.notification("SMS delivered", true);
                break;
            case Activity.RESULT_CANCELED:
                SmsGatewayContainer.notification("SMS not delivered", false);
                break;
        }
    }
}, new IntentFilter("SMS_DELIVERED"));
```

Revisi SmsGatewayReceiver

Terakhir yang perlu kita revisi adalah SmsGatewayReceiver untuk menerima SMS. Kalo untuk ini tinggal kita ubah isi metode onReceive nya.

```
@Override
public void onReceive(Context context, Intent intent) {
    Bundle bundle = intent.getExtras();

    if (bundle != null) {
        // menerima pesan SMS
        Object[] pdus = (Object[]) bundle.get("pdus");
        for (Object pdu : pdus) {
            // conversi ke SmsMessage
            SmsMessage message = SmsMessage.createFromPdu((byte[]) pdu);

            // buat JSON received SMS
            Map<String, Object> map = new HashMap<String, Object>();
            map.put("type", "received");
            map.put("from", message.getOriginatingAddress());
            map.put("message", message.getMessageBody());
            JSONObject response = new JSONObject(map);

            // kirim ke client
            SmsGatewayContainer.send(response.toString());
        }
    }
}
```

Broadcast Message

ngirim ke banyak nomor sekaligus

Broadcast SMS sebenarnya bisa dilakukan di sisi client, tidak perlu dilakukan di sisi server. Cukup dengan melakukan perulangan, misal seperti ini :

```
// kirim sms yang sama ke 100 nomor
for(int i = 0; i < 100; i++) {
    // kirim ke server
    sendToServer(i)
}
```

Tapi masalahnya dengan kode diatas adalah, koneksi harus dilakukan sebanyak 100 kali, artinya proses Network IO nya sangat banyak sekali.

Alangkah baiknya lebih baik cukup kita cantumkan semua nomor yang akan dikirim dalam JSON request, lalu kirim cukup 1 kali. Gak harus berkali-kali, dengan begitu Network IO nya tidak banyak, dan biarkan server yang berkerja :D

Format pesan broadcast message

Untuk melakukan itu, kita perlu mendefinisikan pesan JSON yang akan kita gunakan untuk mengirim SMS broadcast. Berikut adalah contoh JSON yang akan kita gunakan :

```
{  
    "to": [  
        "081111111111",  
        "081111111112",  
        "081111111113",  
        "081111111114",  
        "081111111115",  
        "081111111116"  
    ],  
    "message": "PENGUMUMAN! Ekstrak Kulit Manggis Telah Habis!"  
}
```

Khusus untuk broadcast message, yang sebelumnya atribut "to" bernilai String, sekarang kita ganti menjadi Array of String.

Mengubah onMessage SmsGatewayServer

Karena sekarang ada kemungkinan bahwa isi atribut "to" tidak hanya String, tapi bisa jadi Array of String, maka kita perlu mengubah onMessage yang terdapat pada SmsGatewayServer. Jika data "to" yang diterima adalah Array of String, maka Android SMS Gateway akan melakukan broadcast message ke nomor-nomor tujuan.

```
@Override  
public void onMessage(WebSocket conn, String message) {  
    try {  
        JSONObject object = new JSONObject(message);  
  
        try {  
            JSONArray to = object.getJSONArray("to");  
            String smsMessage = object.getString("message");  
  
            // broadcast message ke semua nomor tujuan  
            for (int i = 0; i < to.length(); i++) {  
                SmsManager.getDefault().sendTextMessage(to.getString(i),  
                    null, smsMessage, sendIntent, deliveryIntent);  
            }  
        } catch (JSONException e) {  
            // to bukan Array of String  
            String to = object.getString("to");  
            String smsMessage = object.getString("message");  
        }  
    } catch (JSONException e) {  
        // to bukan Array of String  
        String to = object.getString("to");  
        String smsMessage = object.getString("message");  
    }  
}
```

```
        SmsManager.getDefault().sendTextMessage(to,
                null, smsMessage, sendIntent, deliveryIntent);
    }

    // kirim pesan sukses
    Map<String, String> map = new HashMap<String, String>();
    map.put("type", "success");
    map.put("message", "Success Send SMS");
    JSONObject response = new JSONObject(map);
    conn.send(response.toString());

} catch (JSONException e) {
    // kirim pesan error
    Map<String, String> map = new HashMap<String, String>();
    map.put("type", "error");
    map.put("message", "Wrong JSON format");
    JSONObject response = new JSONObject(map);
    conn.send(response.toString());
}
}
```

Sekarang Android SMS Gateway yang kita buat sudah bisa menghandle broadcast message jika kita mengirim Array of String pada atribut “to” JSON request-nya.

Aplikasi Desktop

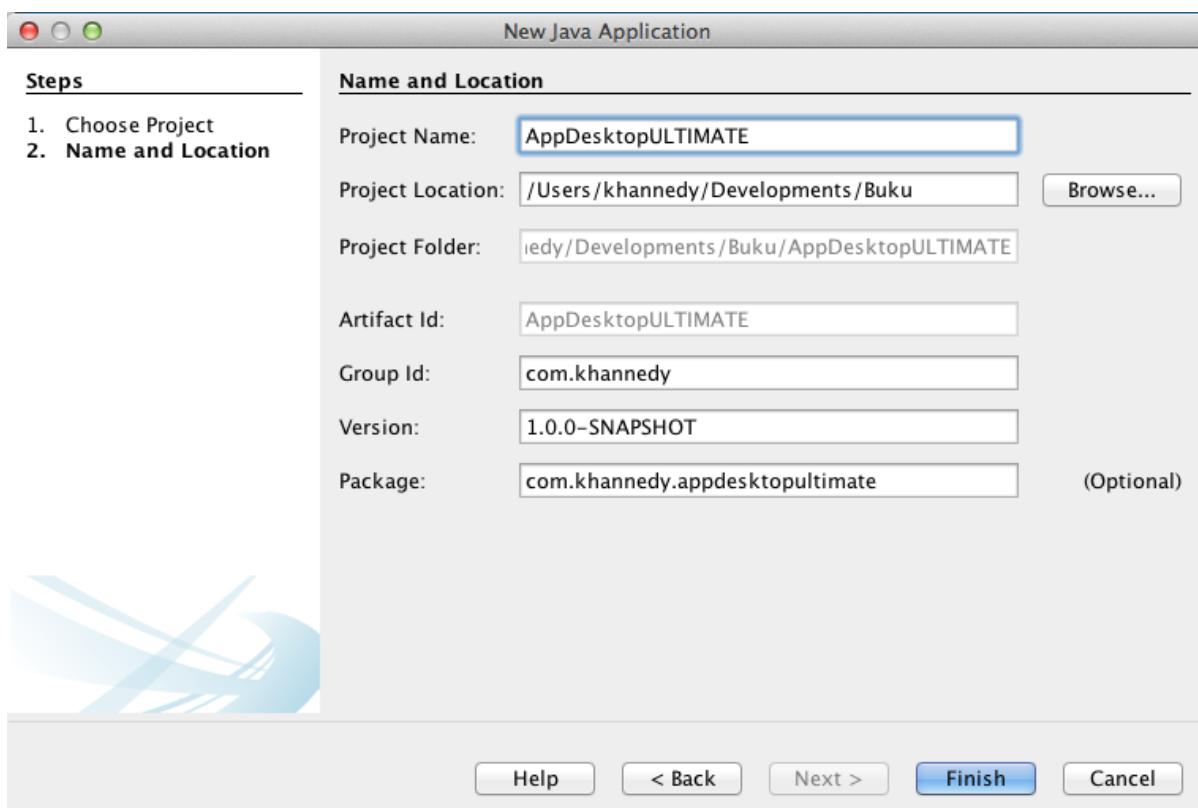
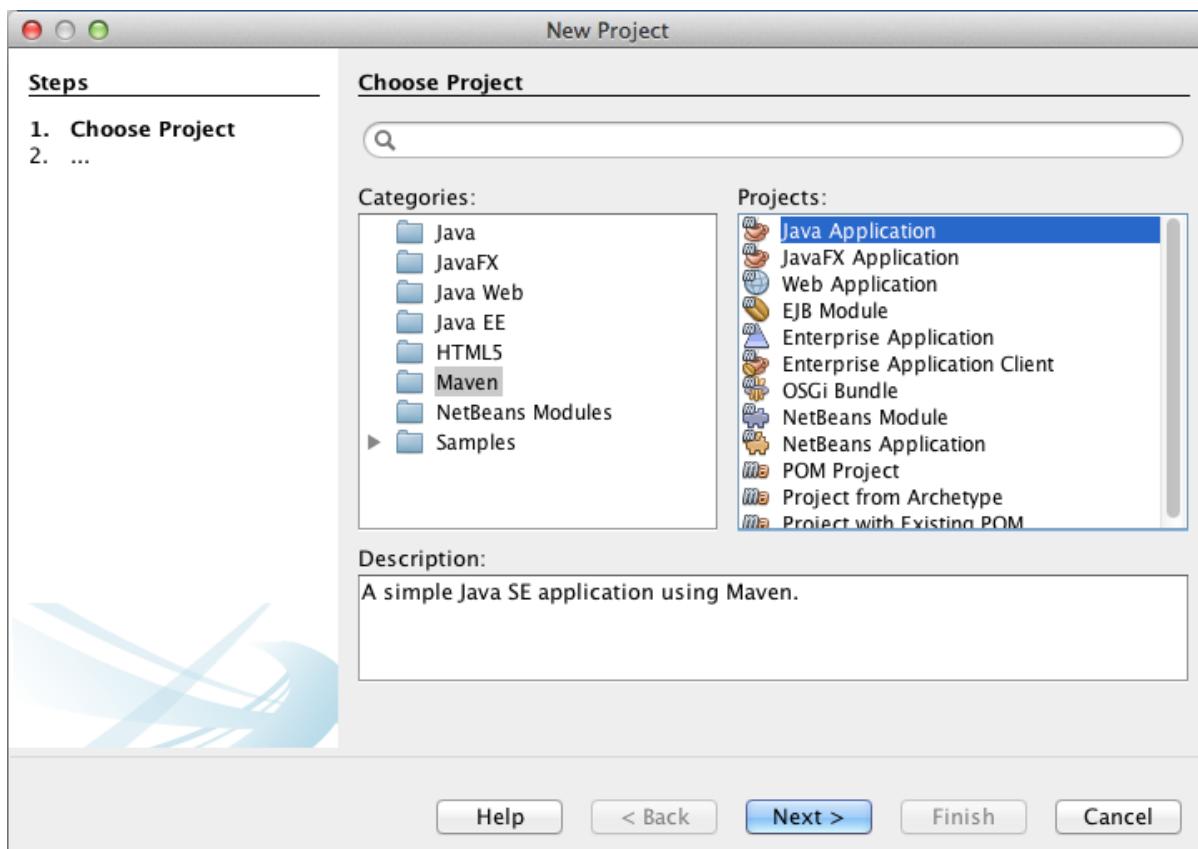
contoh aplikasi desktop

Setelah kita selesai membuat Android SMS Gateway yang ZUPER KEREN! Sekarang saatnya kita fokus membuat aplikasi client yang memanfaatkan Android SMS Gateway-nya. Pada bab ini kita akan fokus untuk membuat aplikasi untuk Desktop, bab selanjutnya kita akan fokus membuat aplikasi untuk Web.

Untuk membuat Aplikasi Desktop-nya, saya akan menggunakan teknologi Java bernama Java Swing. Saran saya Untuk membuat Aplikasi Desktop, gunakanlah NetBeans IDE, kenapa? Karena NetBeans IDE memiliki GUI Builder / Visual Editor yang keren. Jadi kita gak perlu coding manual untuk membuat Form aplikasinya, cukup drag and drop sajah :D

Membuat Project

Pertama, buatlah project Apache Maven Java Application. Silahkan beri nama projectnya dengan nama **AppDesktopULTIMATE**. Sengaja kita akan menggunakan Apache Maven, karena supaya lebih mudah nanti kalo menambahkan library yang kita butuhkan :)



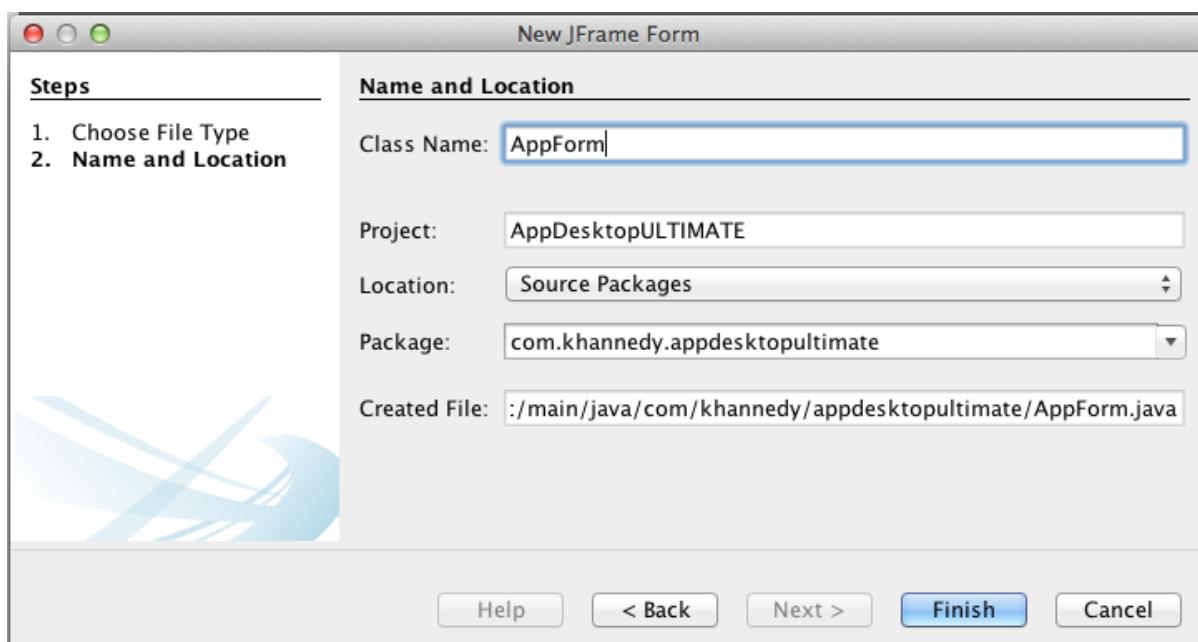
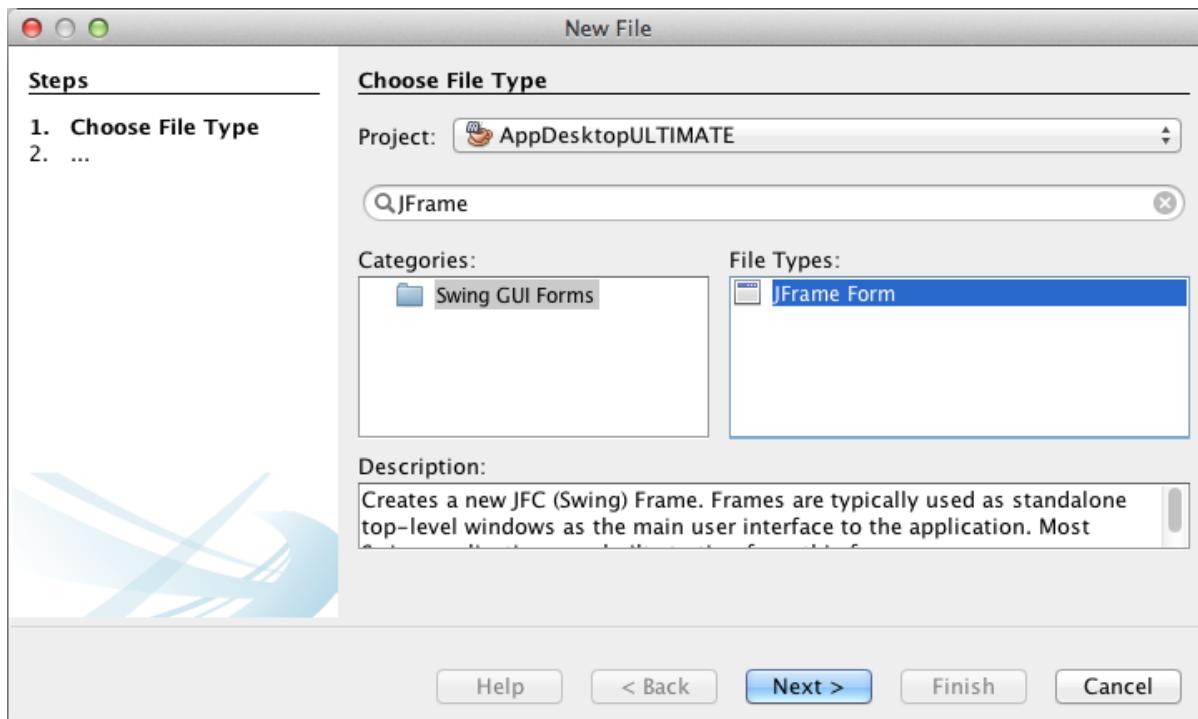
Menambahkan Library

Setelah membuat project, selanjutnya adalah menambahkan library yang kita butuhkan. Library yang kita butuhkan hanyalah **Java-WebSocket** dan juga **Google Gson**. Jadi cukup tambahkan dua library tersebut di **pom.xml**.

```
<dependencies>
    <!--Dependency Java-WebSocket-->
    <dependency>
        <groupId>org.java-websocket</groupId>
        <artifactId>Java-WebSocket</artifactId>
        <version>1.3.0</version>
    </dependency>
    <!--Dependency Google Gson-->
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.3</version>
    </dependency>
</dependencies>
```

Membuat Form Aplikasi

Karena kita akan menggunakan Java Swing, jadi silahkan buat sebuah JFrame form untuk app desktop nya.



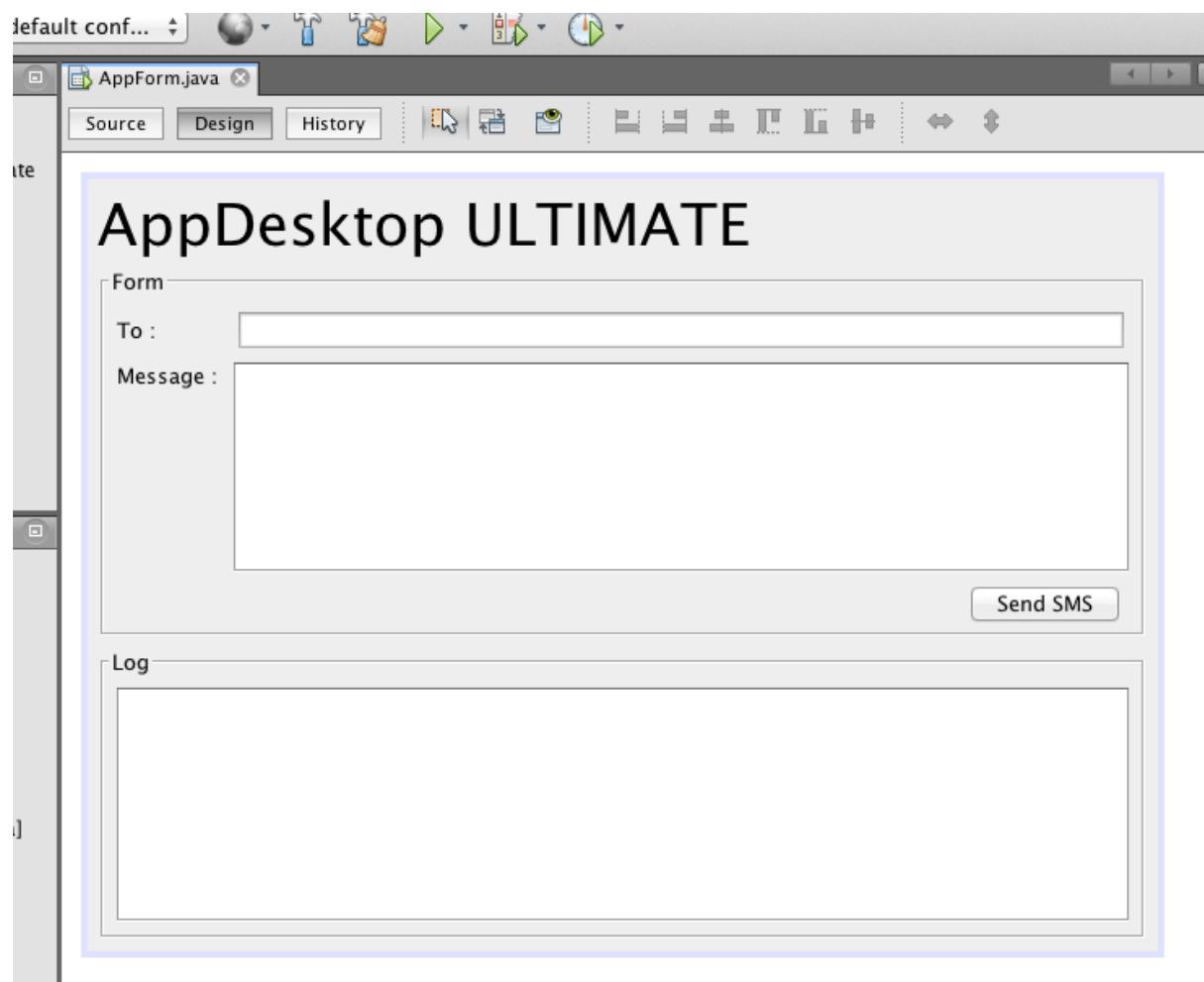
Misal beri nama JFrame nya dengan nama AppForm.

Mendesain Tampilan Form Aplikasi

Kita tidak akan membuat sebuah form aplikasi yang kompleks, sederhana saja, sesuai fungsinya, yaitu memiliki form input untuk no

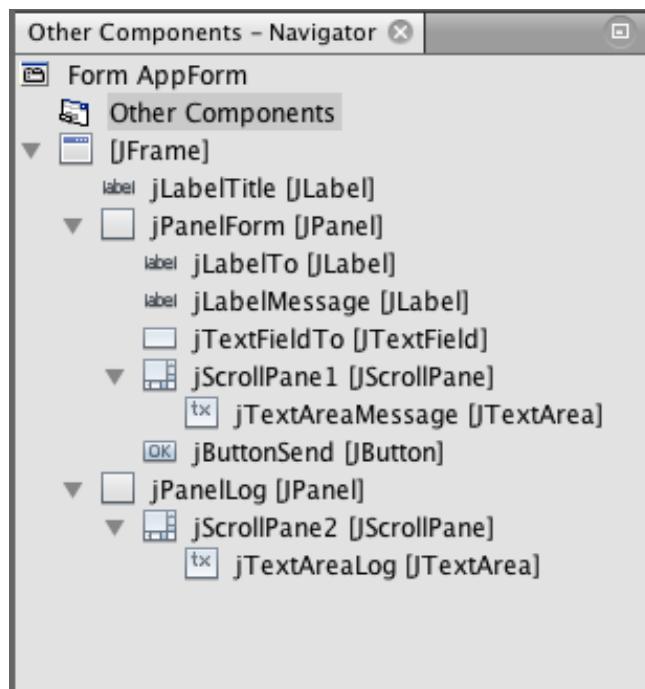
telepon, pesan sms dan tombol kirim sms. Selain itu kita buat sebuah text area untuk log.

Berikut adalah tampilan form yang saya buat, silahkan Anda desain sesuai dengan taste Anda :D



Silahkan ubah nama-nama variabel tiap komponen nya agar mudah dimengerti, secara default, jika kita menambahkan komponen, namanya akan menggunakan numeric auto increment, misal jTextField1, jTextField2. Itu kurang informatif, jadi silahkan ubah nama variabel nya menjadi, misal : jTextFieldNo, jTextAreaMessage, dan lain-lain. Caranya klik kanan komponen-nya lalu pilih **Change Variabel Name**.

Berikut adalah nama-nama variabel yang saya gunakan dalam desain JFrame yang saya buat, kita bisa melihatnya di Navigator NetBeans IDE.



Membuat WebSocketClient

Selanjutnya kita perlu membuat WebSocketClient. Pembuatannya mirip seperti pada contoh di PREMIUM EDITION, kita hanya perlu membuat kelas turunan dari WebSocketClient. Namun sesekarang kita akan mengintegrasikannya dengan Aplikasi Desktopnya, jadi tidak hanya menampilkan tulisan di console saja.

Saat kita sukses atau gagal mengirim SMS, kita akan tampilkan sebuah Message Box. Saat ada notifikasi laporan pengiriman, kita akan tampilkan di Text Area di bagian Log. Dan jika Ada SMS Masuk (Diterima), kita akan tampilkan di Message Box.

Berikut adalah kode WebSocketClient-nya, kita buat dengan nama SmsGatewayClient.

```
package com.khannedy.appdesktopultimate;

import org.java_websocket.client.WebSocketClient;
import org.java_websocket.handshake.ServerHandshake;

import java.net.URI;

/**
 * @author Eko Khannedy
 * @since 12/2/14
 */
public class SmsGatewayClient extends WebSocketClient {

    public SmsGatewayClient(URI serverURI) {
        super(serverURI);
    }

    @Override
    public void onOpen(ServerHandshake handshakedata) {}

    @Override
    public void onMessage(String message) {}

    @Override
    public void onClose(int code, String reason, boolean remote) {}

    @Override
    public void onError(Exception ex) {}
}
```

Membuat WebSocketListener

Karena kelas SmsGatewayClient tidak tergabung dalam kelas AppForm. Maka kita tidak bisa langsung mengintegrasikan SmsGatewayClient dengan AppForm nya. Kita perlu membuat sebuah listener, yang nanti harus di trigger oleh SmsGatewayClient. Listener-nya akan diimplementasikan dalam AppForm.

Sekarang buatlah sebuah **interface** Listener dengan nama WebSocketListener.

```
package com.khannedy.appdesktopultimate;

/**
 * @author Eko Khannedy
 * @since 12/2/14
 */
public interface WebSocketListener {

    void onOpen();

    void onClose();

    void onError();

    void onMessage(String message);

}
```

Isi WebSocketListener hampir mirip dengan metode-metode yang terdapat pada SmsGatewayClient. Kenapa? Karena memang nanti metode-metode di WebSocketListener akan dipanggil oleh SmsGatewayClient dengan nama metode yang sama.

Menambahkan Listener ke SmsGatewayClient

Selanjutnya, setelah membuat Listener, sekarang kita perlu menambahkan Listener tersebut ke dalam SmsGatewayClient-nya. Listener tersebut nanti akan di-supply oleh AppForm.

```
package com.khannedy.appdesktopultimate;

import org.java_websocket.client.WebSocketClient;
import org.java_websocket.handshake.ServerHandshake;

import java.net.URI;

/**
 * @author Eko Khannedy
 * @since 12/2/14
 */
public class SmsGatewayClient extends WebSocketClient {

    private WebSocketListener listener;

    public SmsGatewayClient(URI serverURI, WebSocketListener listener) {
        super(serverURI);
        this.listener = listener;
    }

    @Override
    public void onOpen(ServerHandshake handshakedata) {
        listener.onOpen();
    }

    @Override
    public void onMessage(String message) {
        listener.onMessage(message);
    }

    @Override
    public void onClose(int code, String reason, boolean remote) {
        listener.onClose();
    }

    @Override
    public void onError(Exception ex) {
        listener.onError();
    }
}
```

Sekarang kita telah menambahkan listener ke dalam WebSocketClient. Selanjutnya, kita perlu mensupply listenernya dari AppForm.

Implementasi Listener di AppForm

Agar lebih mudah, dari pada kita buat sebuah class baru yang mengimplementasikan WebSocketListener, lebih baik kita implementasikan langsung ke AppForm-nya. Jadi silahkan tambahkan implements WebSocketListener ke kelas AppForm.

```
/**  
 *  
 * @author Eko Khannedy  
 */  
public class AppForm extends javax.swing.JFrame  
    implements WebSocketListener{
```

Selanjutnya, implementasikan semua metode WebSocketListener di kelas AppForm. Yuk kita implementasikan satu per satu.

Pertama kita implementasikan metode onOpen(). onOpen() akan dipanggil ketika koneksi ke server berhasil (bukan ketika sms berhasil dikirim ya). Jadi kita akan beri log bahwa koneksi ke server berhasil.

```
@Override  
public void onOpen() {  
    jTextAreaLog.setAutoscrolls(true);  
    jTextAreaLog.append("Koneksi ke server berhasil");  
    jTextAreaLog.append("\n");  
}
```

Untuk onClose() kita akan tampilkan status bahwa koneksi ke server terputus.

```
@Override  
public void onClose() {  
    jTextAreaLog.setAutoscrolls(true);  
    jTextAreaLog.append("Koneksi ke server terputus");  
    jTextAreaLog.append("\n");  
}
```

Untuk onError() kita juga akan menampilkan status di Text Area Log.

```
@Override  
public void onError() {  
    jTextAreaLog.setAutoscrolls(true);  
    jTextAreaLog.append("Ups, terjadi error di koneksi");  
    jTextAreaLog.append("\n");  
}
```

Terakhir adalah onMessage(). Implementasi onMessage() lebih panjang dari implementasi metode yang lainnya karena kita harus handle semua kemungkinan jenis pesan yang dikirim oleh server Android SMS Gateway ULTIMATE :)

```
private Gson gson = new Gson();

@Override
public void onMessage(String message) {
    // konversi ke JSON
    JSONObject json = gson.fromJson(message, JSONObject.class);
    if(json.get("type").getAsString().equals("success")){
        // sukses mengirim sms ke server

    }else if(json.get("type").getAsString().equals("error")){
        // gagal mengirim sms ke server

    }else if(json.get("type").getAsString().equals("notification")){
        // menerima status laporan pengiriman sms

    }else if(json.get("type").getAsString().equals("received")){
        // menerima sms dari server
    }
}
```

Dalam metode onMessage() kita melakukan konversi dari String JSON ke object JSONObject milik Google GSON. Selanjutnya kita cek jenis pesan dari atribut "type" nya. Sekarang tinggal kita implementasikan apa yang akan kita lakukan untuk tiap jenis pesan yang diterima dari server.

Untuk jenis "success" dan "error", kita cukup menambahkan sebuah Message Box saja.

```
if(json.get("type").getAsString().equals("success")){
    // sukses mengirim sms ke server
    String msg = json.get("message").getAsString();
    JOptionPane.showMessageDialog(this, msg);

}else if(json.get("type").getAsString().equals("error")){
    // gagal mengirim sms ke server
    String msg = json.get("message").getAsString();
    JOptionPane.showMessageDialog(this, msg);
```

Selanjutnya untuk jenis "notification", kita akan tampilkan di Log TextArea, karena jika ditampilkan di Message Box, terlalu banyak. Karena minimal 1 pengiriman pesan akan mendapatkan status laporan sebagai 2 kali.

```
else if(json.get("type").getAsString().equals("notification")){
    // menerima status laporan pengiriman sms
    String msg = json.get("message").getAsString();
    jTextAreaLog.setAutoscrolls(true);
    if(json.get("success").getAsBoolean()){
        jTextAreaLog.append("Laporan pengiriman sukses : " + msg);
    }else{
        jTextAreaLog.append("Laporan pengiriman gagal : " + msg);
    }
    jTextAreaLog.append("\n");
```

Dan terakhir adalah jenis "received", yaitu ketika ada SMS masuk. Saat ada sms masuk, kita akan menampilkan Confirm Dialog berisikan pesan dan pertanyaan apakah ingin membalas? Jika yup ingin membalas, maka isi jTextFieldNo akan otomatis berisikan no si pengirim SMS.

```
else if(json.get("type").getAsString().equals("received")){
    // menerima sms dari server
    String msg = json.get("message").getAsString();
    String from = json.get("from").getAsString();

    int result = JOptionPane.showConfirmDialog(this, new String[]{
        "SMS dari " + from + " dengan pesan : ",
        msg,
        "Apakah ingin dibalas?"
    });

    if(result == JOptionPane.OK_OPTION){
        // user mengklik OK (balas)

        // ubah text no menjadi no pengirim
        jTextFieldTo.setText(from);
        // kosongkan text message
        jTextAreaMessage.setText("");
    }
}
```

Menjalankan SmsGatewayClient

Kita telah menyelesaikan implementasi dari tiap jenis pesan yang diterima oleh client dari server. Sekarang saatnya kita jalankan SmsGatewayClient dari AppForm.

```
public class AppForm extends javax.swing.JFrame
    implements WebSocketListener{

private SmsGatewayClient client;

/**
 * Creates new form AppForm
 */
public AppForm() {
    initComponents();

    try {
        client = new SmsGatewayClient(new URI("ws://192.168.43.1:8989"), this);
        client.connectBlocking();
    } catch (URISyntaxException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Menambah Aksi tombol Send SMS

Terakhir adalah menambahkan aksi di tombol Send SMS. Caranya klik kanan tombol Send SMS nya lalu pilih **Event -> Action -> actionPerformed**. NetBeans IDE akan otomatis membuatkan metode yang akan diakses ketika tombol Send SMS diklik. Dalam metode ini ayo kita kirim request kirim SMS ke server Android SMS Gateway.

```
private void jButtonSendActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // buat JSON  
    JSONObject object = new JSONObject();  
    object.add("to", new JsonPrimitive(jTextFieldTo.getText()));  
    object.add("message", new JsonPrimitive(jTextAreaMessage.getText()));  
  
    // konversi menjadi String JSON  
    String json = gson.toJson(object);  
  
    // kirim ke server  
    client.send(json);  
  
}
```

Bagaimana dengan Broadcast Message?

Pada form yang kita buat, hanya 1 input no yang kita tambahkan, bagaimana jika kita akan melakukan broadcast message? Gampang saja, kita bisa tambahkan tanda , (koma) pada input no, misal :

08111111111,08111111112,08111111113

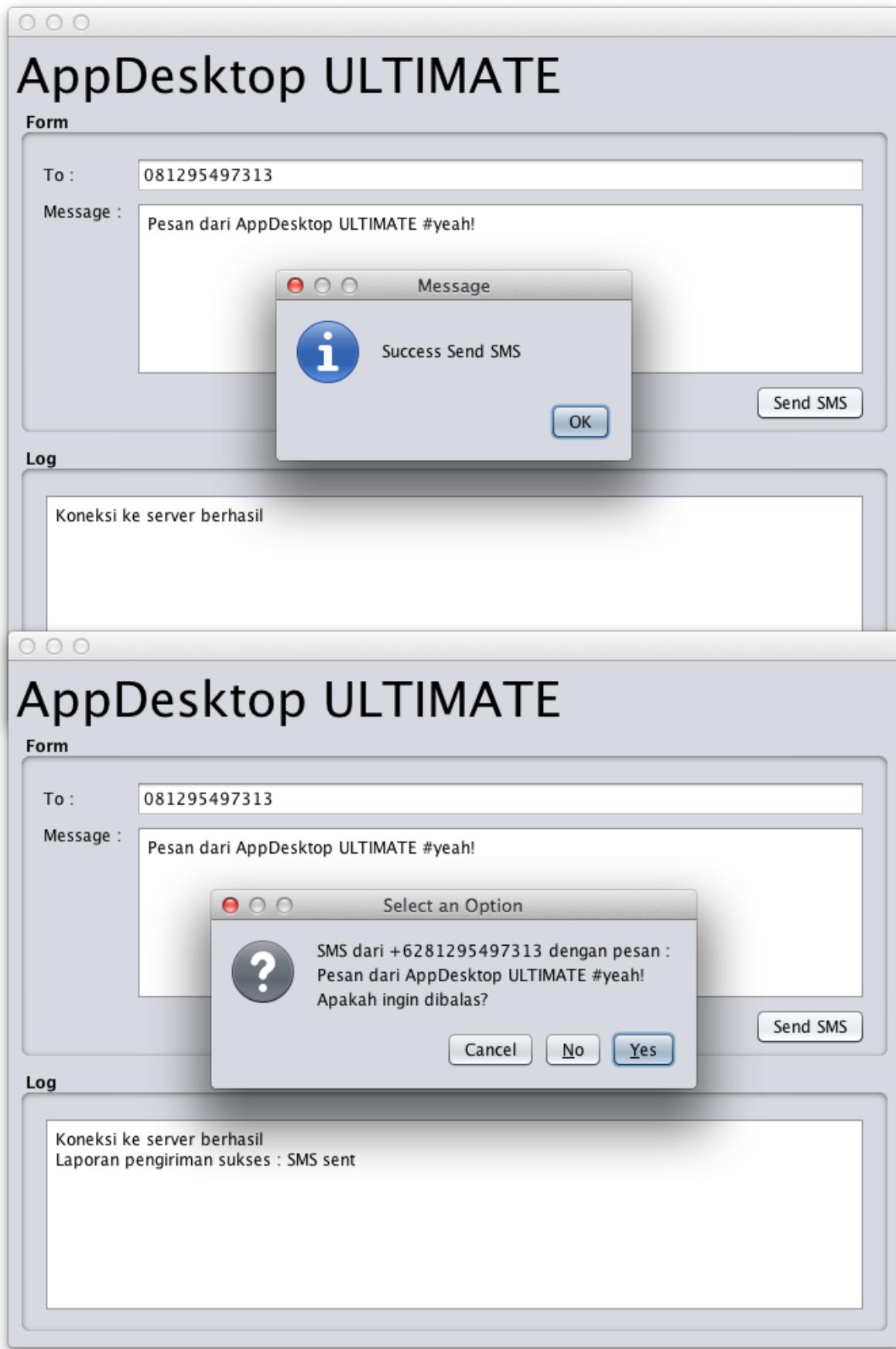
Jadi sekarang kita perlu edit lagi isi metode actionPerformed tombol Send SMS nya.

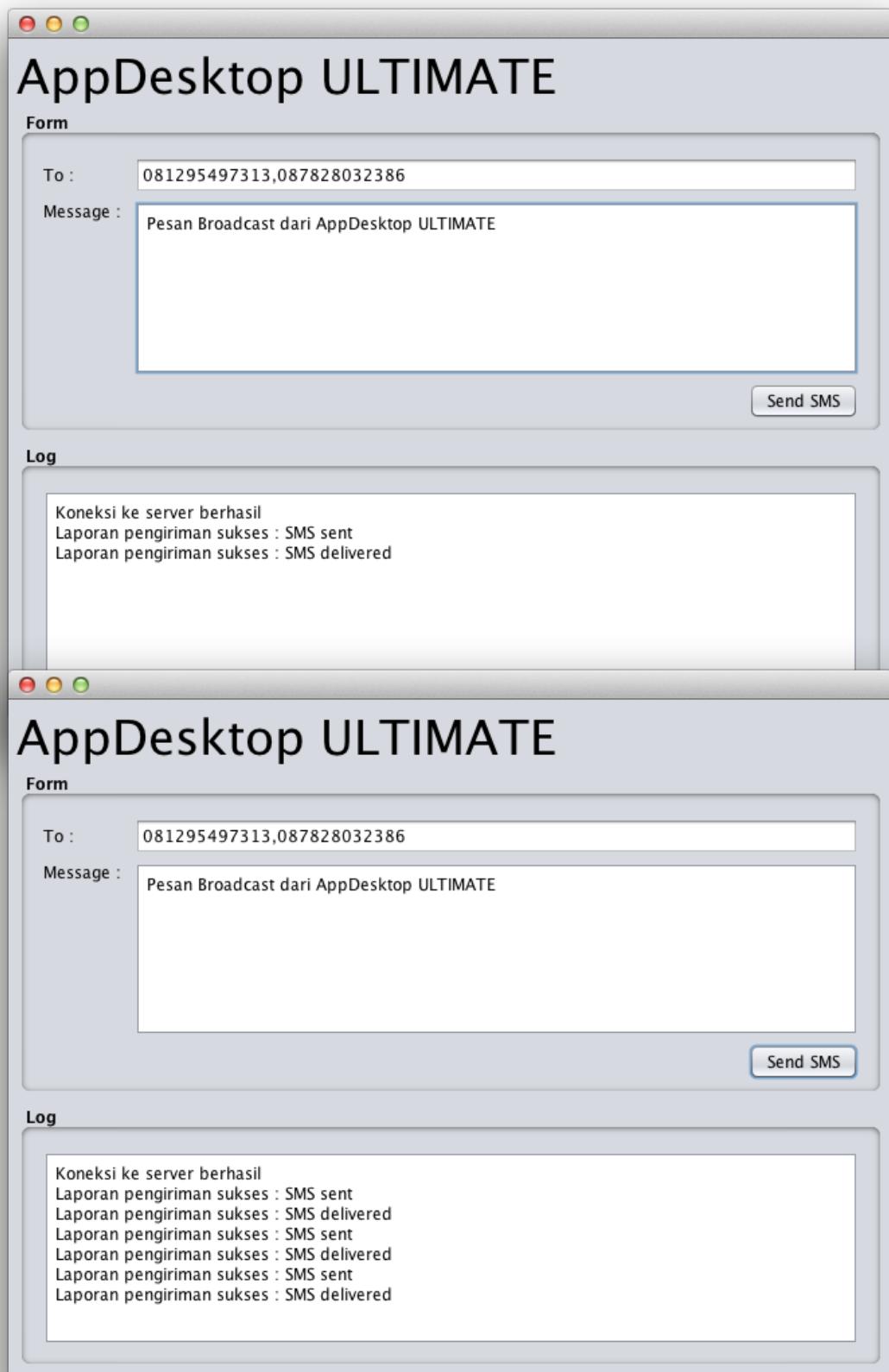
```
private void jButtonSendActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // buat JSON  
    JSONObject object = new JSONObject();  
    object.add("message", new JsonPrimitive(jTextAreaMessage.getText()));  
  
    // cek apakah broadcast message  
    if(jTextFieldTo.getText().contains(",")){  
        // broadcast message  
        String[] tos = jTextFieldTo.getText().split(",");  
        JSONArray array = new JSONArray();  
        for(String to : tos){  
            array.add(new JsonPrimitive(to));  
        }  
        object.add("to", array);  
  
    }else{  
        // bukan broadcast message  
        object.add("to", new JsonPrimitive(jTextFieldTo.getText()));  
    }  
  
    // konversi menjadi String JSON  
    String json = gson.toJson(object);  
  
    // kirim ke server  
    client.send(json);  
}
```

Menguju Aplikasi

Sekarang kita telah selesai membuat Aplikasi Desktop-nya, saatnya kita menguji alias mencobanya. Apakah sudah sesuai dengan apa yang kita inginkan, atau masih ada yang tidak berjalan dengan baik.

Silahkan jalankan projectnya, tinggal klik kanan project-nya lalu pilih Run di NetBeans IDE.





Yeah! SUKSES! Sekarang Anda telah belajar bagaimana membuat aplikasi desktop memanfaatkan Android SMS Gateway ULTIMATE EDITION :D

Diharapkan sekarang Anda tahu cara integrasinya dan bisa diimplementasikan pada Aplikasi Desktop buatan Anda sendiri, misal dengan ini anda sekarang bisa menambah fitur :

- ▶ Notifikasi PENGUMUMAN untuk sistem informasi perpustakaan jika ada buku baru
- ▶ Notifikasi PERINGATAN jika ada pelanggan perpustakaan yang belum mengembalikan buku dan sudah lewat jatuh tempo
- ▶ dan lain-lain :D

Aplikasi Web

contoh aplikasi web

Sekarang zaman-nya web, hampir semua aplikasi mulai berbondong-bondong di migrasikan ke teknologi web. Dahulu sistem informasi banyak yang menggunakan Aplikasi Desktop, namun sekarang lebih banyak yang menggunakan Web.

Bahkan saya sekarang jarang sekali mengerjakan project yang menggunakan Aplikasi Desktop, lebih sering mengerjakan project membuat aplikasi Web.

Di Java, banyak sekali framework untuk Web, ada Java Server Faces, Spring Web MVC, Apache Struts 1 dan 2, Grails, Play Framework dan masih banyak yang lainnya. Lantas mau pakai framework apa sekarang? Hmm, jika saya menggunakan salah satu framework yang saya sebutkan tadi, saya takutnya ada pembaca yang tidak familias dengan framework tersebut. Atau mungkin bisa jadi framework nanti yang saya pilih, malah tidak disukai oleh sabagian pembaca.

Oleh karena itu, pada Aplikasi Web ini, saya akan putuskan bahwa kita tidak akan menggunakan framework apapun. Hah? Trus? Manual pake Servlet? Enggak juga. Bukankah WebSocket itu teknologi HTML5? Lantas kenapa kita harus menggunakan framework web server? Lebih baik kita gunakan HTML5 saja alias menggunakan JavaScript!

Jika Anda membuat aplikasi web, sudah pasti Anda akan bertemu dengan HTML + JavaScript. Dengan begitu framework apapun yang Anda suka, maka tetap Anda wajib mengerti HTML + JavaScript :D

Membuat Project Web

Yuk kita mulai dari membuat projectnya terlebih dahulu. Karena kita menggunakan HTML + JavaScript, jadi Anda boleh menggunakan IDE, atau bahkan menggunakan Text Editor seperti Sublime, NodePad ++ atau bahkan Vim.

Cukup buatkah sebuah folder dengan nama **aplikasi-web-ULTIMATE**. Lalu buat 2 buah file, 1 file **index.html** dan 1 lagi file **app.js**.

```
<!DOCTYPE html>
<html>
<head>
    <title>Aplikasi Web ULTIMATE</title>
</head>
<body>
<h1>Aplikasi Web ULTIMATE</h1>
<script type="text/javascript" src="app.js"></script>
</body>
</html>
```

index.html akan berisikan form aplikasi untuk mengirim SMS, dan app.js akan berisikan kode program untuk melakukan request WebSocket ke Android SMS Gateway ULTIMATE.

Membuat Form Web

Form yang akan kita buat tidak lah kompleks, kita akan meniru form yang terdapat pada Aplikasi Desktop. Kita akan buat sebuah input-text untuk no tujuan, textarea untuk pesan SMS, button untuk Send SMS dan terakhir adalah textarea untuk log.

Saya sarankan untuk Anda, silahkan buat formnya yang bagus menggunakan CSS framework seperti bootstrap atau yang lainnya.

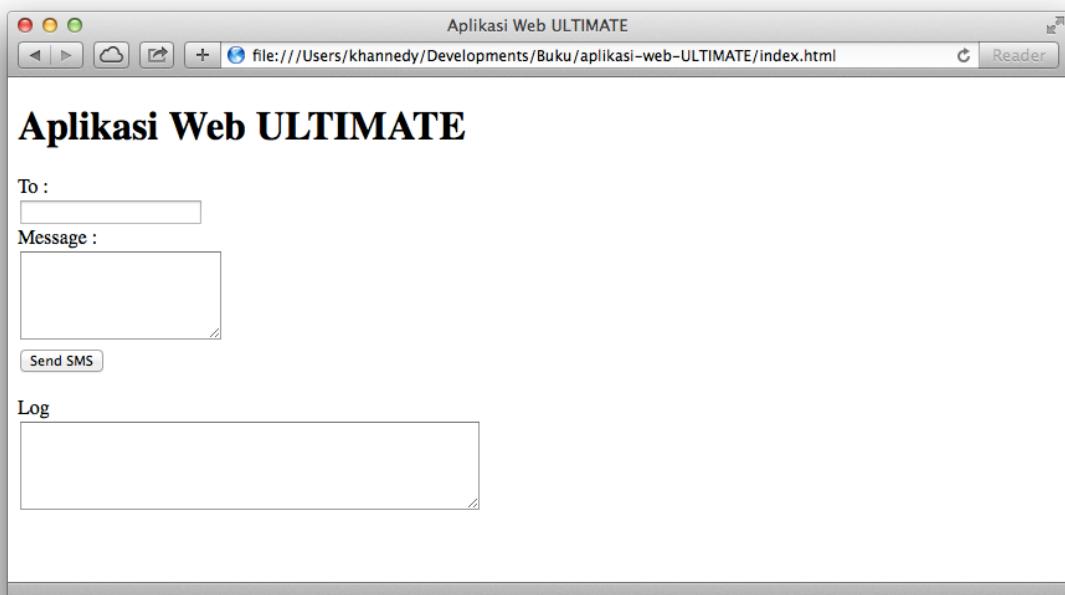
Kalo yang sekarang saya buat tidak akan menggunakan CSS framework, sehingga tampilannya ala kadarnya :D

```
<!DOCTYPE html>
<html>
<head>
    <title>Aplikasi Web ULTIMATE</title>
</head>
<body>
<h1>Aplikasi Web ULTIMATE</h1>

<label for="to">To :</label> <br/>
<input type="text" id="to"/> <br/>
<label for="message">Message :</label> <br/>
<textarea id="message" cols="20" rows="5"></textarea> <br/>
<button id="send">Send SMS</button>

<br/><br/>
<label for="log">Log</label> <br/>
<textarea id="log" cols="50" rows="5"></textarea> <br/>

<script type="text/javascript" src="app.js"></script>
</body>
</html>
```



Form aplikasi web telah kita buat, sekarang saat membuat logic program nya menggunakan JavaScript. Logic program kita buat dalam file app.js. Jadi kita pisahkan antara View (HTML) dan logic (JavaScript) nya.

```
function startApp() {  
  
    // create websocket client  
    var client = new WebSocket("ws://192.168.43.1:8989");  
  
    // onOpen handler  
    client.onopen = function (event) {  
        var log = document.getElementById("log");  
        log.textContent = log.textContent + "\n" + "Koneksi ke server berhasil";  
    };  
  
    // onClose handler  
    client.onclose = function (event) {  
        var log = document.getElementById("log");  
        log.textContent = log.textContent + "\n" + "Koneksi ke server terputus";  
    };  
  
    // onError handler  
    client.onerror = function (event) {  
        var log = document.getElementById("log");  
        log.textContent = log.textContent + "\n" + "Koneksi ke server error";  
    };  
  
}  
  
window.onload = startApp;
```

Dalam kode diatas, kita telah membuat sebuah koneksi ke WebSocket menggunakan JavaScript. Selanjutnya kita tambahkan beberapa event ketika **onopen**, **onclose** dan **onerror**. Yang belum kita implementasikan adalah **onmessage**. Nanti kita implementasikan onmessage setelah kita tambahkan aksi untuk mengirim SMS dari tombol Send SMS.

Menambah Aksi Send SMS

Sekarang kita tambahkan event onclick pada tombol Send SMS untuk mengirimkan request ke websocket server.

```
// aksi tombol Send SMS
document.getElementById("send").onclick = function () {
    // mengambil value no tujuan
    var to = document.getElementById("to").value;
    // mengambil value isi pesan sms
    var message = document.getElementById("message").value;

    // membuat json
    var json = {
        to: to,
        message: message
    };

    // mengirim ke server via websocket
    client.send(JSON.stringify(json));
}
```

Sekarang kita telah mengimplementasikan aksi onclick tombol Send SMS. Selanjutnya kita perlu tambahkan handler untuk onmessage pada WebSocket Client.

Implementasi onmessage WebSocket Client

Dalam handler onmessage, kita akan mendapatkan response data dari server Android SMS Gateway berupa JSON dengan berbagai macam tipe pesan. Oleh karena itu untuk onmessage, handler nya cukup panjang. Dan akan kita pecah menjadi beberapa pembahasan berdasarkan tipe pesan nya.

Sebelumnya kita buat dulu draft handler onmessage di JavaScript nya.

```
// onMessage handler
client.onmessage = function (event) {
    var response = JSON.parse(event.data);

    switch (response.type) {
        case "success":
            // sukses mengirim sms ke server
            break;

        case "error" :
            // gagal mengirim sms ke server
            break;

        case "notification" :
            // laporan status pengiriman sms
            break;

        case "received" :
            // menerima sms
            break;
    }
};
```

Pertama kita akan implementasikan tipe pesan "success" dan "error". Saat menerima jenis pesan ini, kita akan menampilkan Message Box.

```
switch (response.type) {
    case "success":
        // sukses mengirim sms ke server
        alert(response.message);
        break;

    case "error" :
        // gagal mengirim sms ke server
        alert(response.message);
        break;
```

Selanjutnya jika mendapatkan tipe "notification", kita akan tampilkan informasinya di textarea log.

```
case "notification" :  
    // laporan status pengiriman sms  
    var log = document.getElementById("log");  
    if (response.success) {  
        log.textContent = log.textContent + "\n" +  
            "Laporan sukses : " + response.message;  
    } else {  
        log.textContent = log.textContent + "\n" +  
            "Laporan gagal : " + response.message;  
    }  
    break;
```

Terakhir adalah implementasi untuk tipe "received" yang artinya ada SMS Masuk. Jika ada SMS masuk, kita akan tampilkan konfirmasi apakah akan membalas pesan, jika Ya, maka no pengirim SMS akan dimasukkan dalam input text no tujuan.

```
case "received" :  
    // menerima sms  
    if (confirm("Sms dari " + response.from + " : \n" +  
        response.message + "\n" +  
        "Apakah ingin dibalas?")) {  
        document.getElementById("to").value = response.from;  
    }  
    break;
```

Sekarang kita telah mengimplementasikan semua handler untuk WebSocket client-nya. Selanjutnya kita implementasikan fitur yang terakhir yaitu Broadcast Message.

Broadcast Message

Karena input no tujuan hanya 1, maka kita akan gunakan separator , (koma) untuk pemisah jika user ingin mengirim ke nomor lebih dari 1, misal : 081111111111,081111111112,081111111113.

Dalam kodennya berarti kita perlu mendeteksi apakah terdapat lebih dari 1 nomor, jika lebih dari satu nomor, maka kita perlu mengirimnya secara broadcast, namun jika tidak, maka kita tidak perlu membuat request broadcast.

```
// aksi tombol Send SMS
document.getElementById("send").onclick = function () {
    // mengambil value no tujuan
    var to = document.getElementById("to").value;
    // mengambil value isi pesan sms
    var message = document.getElementById("message").value;

    var splits = to.split(",");
    if (splits.length == 1) {
        // bukan broadcast

        // membuat json
        var json = {
            to: splits[0],
            message: message
        };

        // mengirim ke server via websocket
        client.send(JSON.stringify(json));

    } else {
        // broadcast

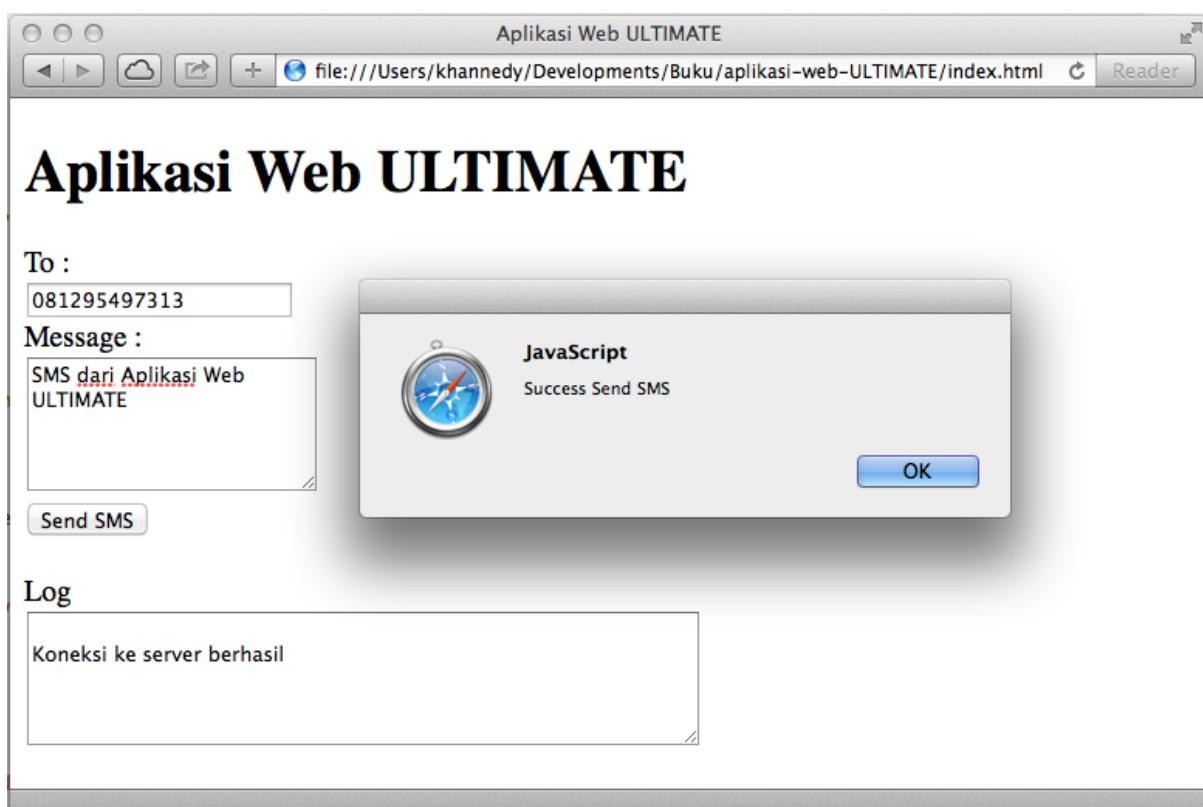
        // membuat json broadcast
        var json = {
            to: splits,
            message: message
        };

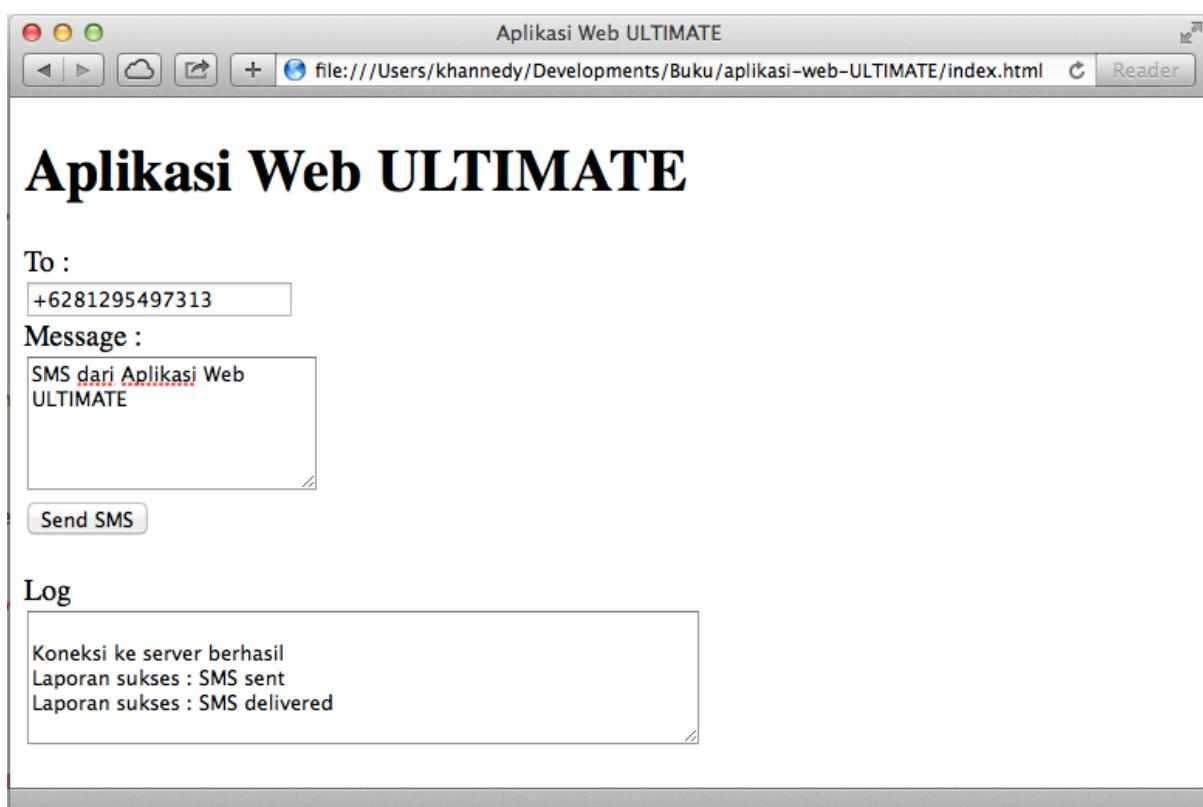
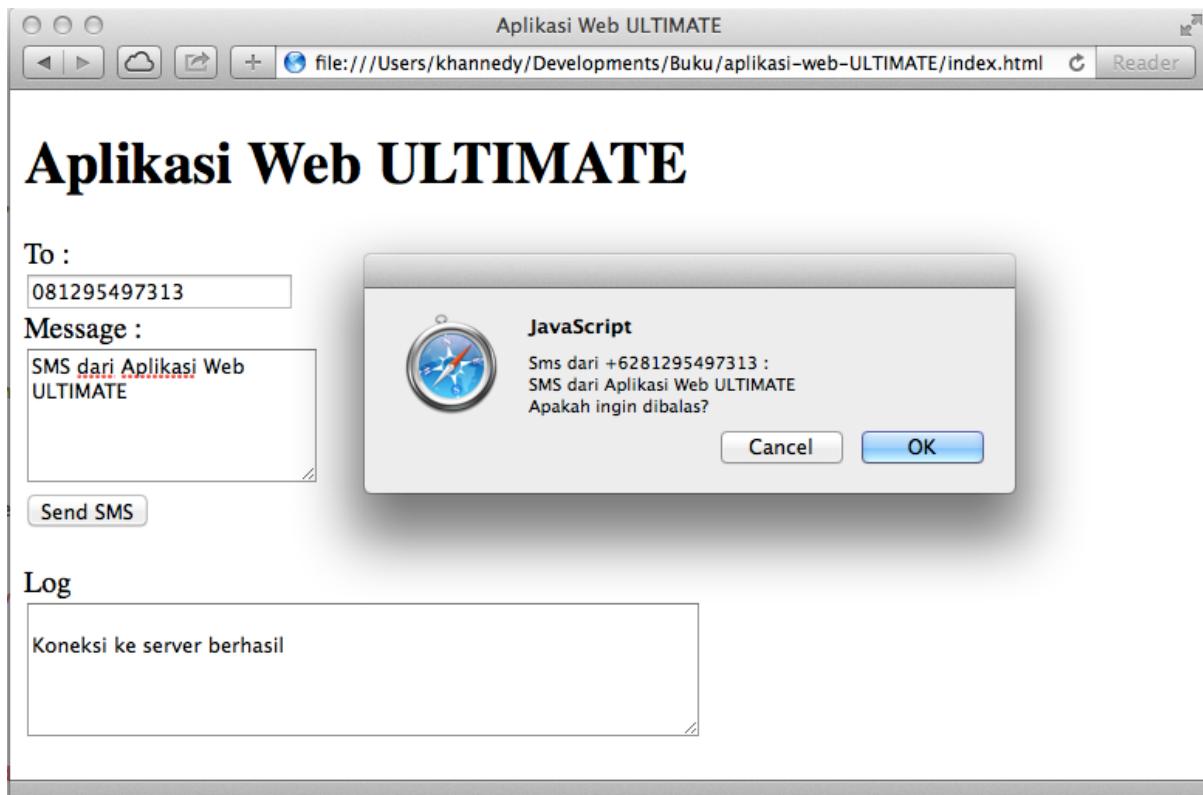
        // mengirim ke server via websocket
        client.send(JSON.stringify(json));
    }
}
```

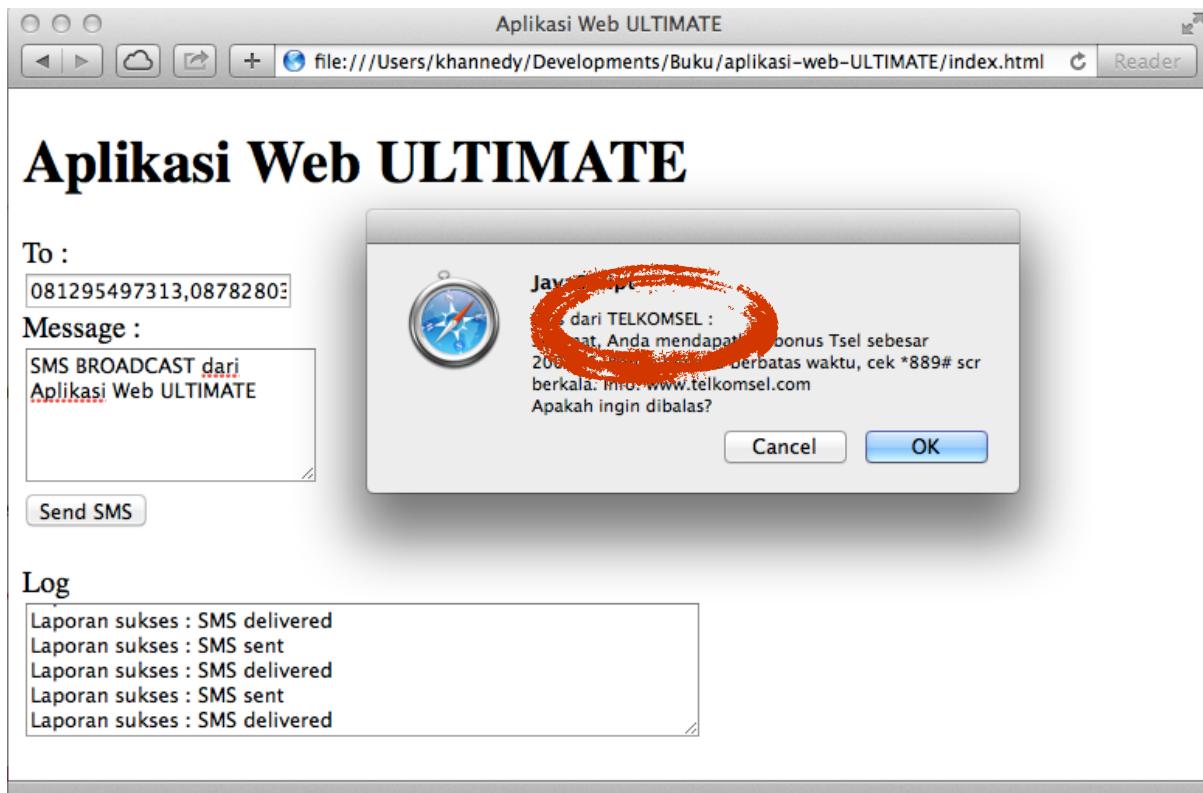
Sekarang kita telah selesai membuat Aplikasi Web untuk Android SMS Gateway ULTIMATE. Selanjutnya tahapan terakhir adalah mengujinya apakah sudah sesuai dengan yang kita inginkan atau belum Aplikasi Web nya :D

Menguji Aplikasi

Sekarang kita uji coba aplikasinya. Berikut adalah hasil uji coba yang saya lakukan.







Bahkan TELKOMSEL juga malah ikut-ikutan ngetest aplikasi Android SMS Gateway ULTIMATE nya :D #hehehe