Subang
November 27, 2014

# Membuat Aplikasi Chat Menggunakan Java

Versi 1.1

```
package com.khannedy;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class Welcome {

    public static void main(String[] args) {
        String welcome = "Selamat Datang di Seri Tutorial Java " +
                "Bersama Eko Kurniawan Khannedy";
        System.out.println(welcome);
    }

}
```


Eko Khannedy

@khannedy

khannedy.com

# Daftar Isi

## apa aja isi bukunya

# Persiapan

## yang perlu dilakukan sebelum bertarung

Sebelum bertarung (membuat aplikasi chat), ada beberapa hal yang perlu kita lakukan, seperti menentukan aplikasi chat seperti apa yang akan dibuat dan juga peralatan apa saja yang akan kita gunakan untuk membuat aplikasi chat.

### Spesifikasi aplikasi

- Aplikasi chat berjalan diatas terminal / command prompt.
- Server dapat menghandle lebih dari 1 client secara bersamaan.
- Client dapat berinteraksi dengan semua client lainnya.
- Client perlu memilih username terlebih dahulu sebelum bisa berchating.

### Peralatan yang dibutuhkan

- Java Development Kit 8 (*kita akan menggunakan fitur Java 8 Stream API dan Lambda Expression*)
- IDE (*bisa menggunakan IntelliJ IDEA, NetBeans atau Eclipse*)
- Build Tool (*disini menggunakan Apache Maven*)
- Netty (*untuk Network Programming Framework*)
- Google Gson (*untuk JSON Converter*)

# Konfigurasi Project

## menggunakan Apache Maven

Buatlah project menggunakan Apache Maven lewat IDE, hampir semua IDE modern sekarang (IntelliJ IDEA, NetBeans dan Eclipse) sudah mendukung build tool Apache Maven.

### Ubah Source dan Target Project ke Java 8

Secara default, Apache Maven akan mengkompilasi kode program agar kompetibel dengan Java 6, dengen begitu, fitur Java 8 tidak bisa digunakan jika walaupun kita menggunakan JDK 8, jadi pastikan set compiler dan source version ke Java 8, tambahkan kode dibawah ini di file **pom.xml** project yang sudah dibuat.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="...">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.khannedy</groupId>
    <artifactId>aplikasi-chating</artifactId>
    <version>1.0.0-SNAPSHOT</version>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.2</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Source dan Target 1.8 pada kode XML diatas digunakan agar ketika proses kompilasi, Apache Maven akan menganggap kode program dibuat menggunakan Java 8 sehingga fitur Java 8 otomatis bisa digunakan. Namun perlu diketahui, dengan begitu, program Java nya juga hanya bisa dijalankan oleh Java 8 atau versi diatasnya.

## Menambah Library Netty

Sebelumnya sudah saya bahas bahwa kita akan menggunakan Netty untuk Network Programming Framework nya. Kita tidak akan manual menggunakan Java Network API karena hal itu cukup lama pembuatannya. Saya menggunakan Netty karena sampai saat ini Netty masih menjadi framework nomor 1 untuk network programming.

Netty secara otomatis menggunakan Event Driven, secara awam ini artinya Netty menghandle client secara **non-blocking**. Sebuah aplikasi dikatakan **blocking** jika ada 2 proses atau lebih, proses yang datang belakangan harus menunggu proses yang lebih awal dulu selesai, baru bisa di proses. Netty tidak demikian, dari awal Netty memang didesain untuk Non-Blocking Network Programming.

Untuk menambahkan library Netty, tambahkan sebuah dependency baru di **pom.xml** project Apache Maven nya.

```
<dependencies>

    <dependency>
        <groupId>io.netty</groupId>
        <artifactId>netty-all</artifactId>
        <version>4.0.24.Final</version>
    </dependency>

</dependencies>
```

## Menambah Library Google GSON

Hei, bukannya kita akan membuat aplikasi chat? Bukan aplikasi web? Kenapa kita perlu Google GSON? Bukankah Google GSON itu framework untuk JSON Converter?

Yup, semua pertanyaan itu benar, tidak ada yang salah :D Lantas untuk apa Google GSON? Google GSON atau lebih biasa disebut GSON, akan kita gunakan untuk mengkonversi objek Java menjadi String JSON. Pada aplikasi chat yang akan kita buat nanti, data yang dikirim dari client ke server dan juga sebaliknya dari server ke client adalah data berupa String JSON.

Kenapa String JSON? Kenapa tidak String plain text? Jika kita menggunakan String plain text, akan sangat sulit jika kita mengirim informasi di dalam data yang dikirim. Namun akan sangat mudah jika kita menggunakan String JSON. Misal sebagai berikut.

```
{
  "sender" : "Eko Khannedy",
  "message" : "Hai apa kabar?",
  "datetime" : "27-11-2014 12:04:09"
}
```

Dengan menggunakan String JSON diatas, kita bisa dengan mudah tau jika ada pesan yang datang dikirim oleh **“Eko Khannedy”** dengan pesan **“Hai apa kabar?”** pada waktu **"27-11-2014 12:04:09”**, namun jika kita menggunakan String Plain Text, maka akan sulit dimengerti, karena informasi bergabung dengan pesan aslinya.

```
"Eko Khannedy Hai apa kabar? 27-11-2014 12:04:09”
```

Untuk menambahkan library GSON silahkan tambahkan kode dibawah ini pada file **pom.xml** project Apache Maven nya.

```
<dependencies>
    <dependency>
        ..... // dependency Netty
    </dependency>

    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.3.1</version>
    </dependency>
</dependencies>
```

Sekarang kita telah selesai melakukan konfigurasi project Aplikasi Chat kita, saatnya kita buat Aplikasi Chat nya :D #semangat!!!!

# Membuat Base Project

## kode yang digunakan Server dan Client

Sebelum mulai membuat aplikasi Server dan Client, terlebih dahulu perlu dibuat kode yang akan digunakan baik itu di Server dan juga Client.

Kode apa saja yang digunakan Server dan Client?

- Format Pesan JSON
- JSON Converter

### Format Pesan JSON

Format pesan JSON yang akan kita gunakan adalah sesuai dengan keperluan aplikasi chat kita, berikut adalah format data yang dibutuhkan :

- **sender**, data pengirim pesan chat
- **message**, data pesan chat
- **datetime**, waktu mengirim pesan
- **action**, aksi yang dilakukan oleh pengirim (***login, chat, logout***)

Format data diatas perlu kita buat menjadi sebuah kelas, dan dengan memanfaatkan GSON, kita bisa konversi objek kelas nya menjadi String JSON sebelum dikirim via Netty.

### Membuat ENUM ChatAction

Format JSON akan kita representasikan dalam sebuah kelas Java, namun sebelumnya kita akan membuat ENUM untuk atribut **action**. Kenapa dibuat ENUM tidak String? Karena jika kita menggunakan String, bisa saja terjadi kesalahan saat mengubah atribu action, misal action = "*logoout*", seharusnya "*logout*". Jika kita menggunakan ENUM makan tidak mungkin ada salah typo.

```
package com.khannedy.chating;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public enum ChatAction {
    LOGIN, LOGOUT, CHAT
}
```

## Membuat Kelas ChatMessage

Setelah membuat ENUM untuk action dengan nama Chat Action, selanjutnya kita buat kelas ChatMessage sebagai representasi Format JSON yang akan digunakan oleh Server dan Client.

```
package com.khannedy.chating;

import java.util.Date;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ChatMessage {

    private String sender; // informasi pengirim
    private String message; // pesan yang dikirim
    private ChatAction action; // jenis aksi (login, logout, chat)
    private Date datetime; // waktu pengiriman pean
```

```
    public String getSender() { return sender; }

    public void setSender(String sender) { this.sender = sender; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

    public ChatAction getAction() { return action; }

    public void setAction(ChatAction action) { this.action = action; }

    public Date getDatetime() { return datetime; }

    public void setDatetime(Date datetime) { this.datetime = datetime; }
}
```

## Membuat ChatMessageConverter

Selanjutnya yang perlu kita lakukan adalah membuat sebuah kode yang dapat digunakan untuk mengubah objek ChatMessage menjadi String JSON dan juga sebaliknya dari String JSON menjadi objek ChatMessage.

Seperti pembahasan sebelumnya, kita akan menggunakan GSON, namun supaya lebih mudah kita perlu membuat kelas utilitas, misal dengan nama ChatMessageConverter, dengan tujuan supaya proses conversi dari String JSON ke objek ChatMessage menjadi lebih mudah, hanya cukup menggunakan kode seperti dibawah ini.

```
// mengubah dari objek ChatMessage ke string JSON
String json = ChatMessageConverter.convertToJSON(chatMessage);

// mengubah dari string JSON ke objek ChatMessage
ChatMessage chatMessage = ChatMessageConverter.convertFromJSON(json);
```

Kelas ChatMessageConverter yang akan kita buat adalah kelas utilitas, jadi kita hanya perlu membuat metode static di kelas ChatMessageConverter.

```
package com.khannedy.chating;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ChatMessageConverter {

    private static Gson gson = new GsonBuilder().
            setDateFormat("dd/MM/yyyy HH:mm:ss").
            create();

    public static String convertToJSON(ChatMessage chatMessage) {
        return gson.toJson(chatMessage);
    }

    public static ChatMessage convertFromJSON(String json) {
        return gson.fromJson(json, ChatMessage.class);
    }
}
```

Oke, sepertinya sekarang kita sudah selesai membuat base project yang akan digunakan baik itu oleh Server maupun Client. Bab selanjutnya kita akan mulai membuat aplikasi chat Server dan dilanjutkan dengan aplikasi chat Client di bab setelahnya #tetapSemangat

# Aplikasi Chat Server

## bertugas untuk menghandle client

Aplikasi Chat Server bertugas untuk menerima client dan mendistribusikan pesan yang dikirim oleh client ke client yang lainnya. Tanpa aplikasi Server, tidak mungkin Client bisa merkomunikasi dengan Client lain.

### Membuat ClientDatabase

Jangan kaget dengan judul "database", karena bukan berarti kita akan menggunakan sistem database. Supaya server bisa mengirim pesan dari satu client ke client yang lainnya, maka server perlu menyimpan data client. Dan untuk melakukan tugas ini, kita akan membuat sebuah kelas ClientDatabase, yang bertugas untuk menyimpan data client dan juga mendistribusikan pesan dari satu client ke client yang lainnya.

```
package com.khannedy.chating;

import io.netty.channel.Channel;

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

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ClientDatabase {

    /**
     * io.netty.channel.Channel merupakan representasi dari client
     * di framework Netty
     */
    private static List<Channel> channels = new ArrayList<>();
```

Ada beberapa operasi database yang akan kita tambahkan, yaitu;

- **login**, menambahkan client ke database ketika client login
- **logout**, menghapus client dari database ketika client logout
- **chat**, mendistribusikan pesan dari client ke client yang lainnya

Mari kita buat semua operasinya di kelas ChatDatabase.

## Membuat Operasi login()

```
/**
 * Menambah client ke databae ketika login
 *
 * @param channel chat client
 */
public static void login(Channel channel) {
    channels.add(channel);
}
```

Untuk menambahkan client ke database ketika login, sekarang kita bisa menggunakan kode berikut

```
// menambah client ke database ketika login
ClientDatabase.login(channel);
```

## Membuat Operasi logout()

```
/**
 * Menghapus client dari database ketika logout
 *
 * @param channel chat client
 */
public static void logout(Channel channel) {
    channels.remove(channel);
}
```

Untuk menghapus client ketika logout, sekarang kita bisa menggunakan kode berikut.

```
// menghapus client dari database ketika logout
ClientDatabase.logout(channel);
```

## Membuat Operasi chat()

Terakhir yang perlu kita buat adalah operasi chat() untuk mengirim pesan chating dari client ke client yang lain.

```
/**
 * Mengirim pesan dari client ke client yang lain
 *
 * @param sender client pengirim pesan
 * @param json   pesan yang dikirim
 */
public static void chat(Channel sender, String json) {
    // stream dan lambda expression adalah fitur java 8
    channels.stream()
            // filter untuk hapus pengirim supaya tidak dikirim pesan nya
            .filter(channel -> channel != sender)
            // kirim kesemua client selain pengirim pesan
            .forEach(channel -> channel.writeAndFlush(json));
}
```

Pada operasi chat() diatas, saya menggunakan fitur Java 8 yaitu Stream dan Lambda Expression supaya lebih mudah dan cepat. Inti dari operasi diatas adalah, filter data sender (pengirim) dari database, setelah itu kirim data pesan nya ke seluruh client yang telah di filter, dengan begitu, client yang mengirim pesan (sender) tidak akan mendapatkan broadcast pesan dia sendiri.

Kode diatas jika **tanpa menggunakan fitur Stream dan Lambda Expression Java 8**, kodenya akan seperti berikut ini.

```
/**
 * Mengirim pesan dari client ke client yang lain
 *
 * @param sender client pengirim pesan
 * @param json   pesan yang dikirim
 */
public static void chat(Channel sender, String json) {

    // operasi chat() tanpa menggunakan stream dan lambda expression java 8

    // filter untuk hapus pengirim supaya tidak dikirim pesan nya
    List<Channel> hasilFilter = new ArrayList<>();
    for (Channel channel : channels) {
        if (channel != sender) {
            // jika bukan sender, tambahkan ke hasilFilter
            hasilFilter.add(channel);
        }
    }

    // kirim kesemua client selain pengirim pesan
    for (Channel channel : hasilFilter) {
        channel.writeAndFlush(json);
    }
}
```

Itulah kenapa saya menggunakan Stream dan Lambda Expression di Java 8, karena kodenya akan lebih singkat dan mudah dibaca. Untuk mengirim broadcast pesan dari client ke client yang lainnya, sekarang kita bisa menggunakan kode sebagai berikut.

```
// mengirim pesan ke client lain
ClientDatabase.chat(channel, json);
```

## Membuat ServerListener

Seperti yang sudah dibawah sebelumnya, jika Netty adalah sebuah Network Programming Framework yang berbasis Event-Driven, dengen begitu, semua kita tidak perlu menghandle koneksi client secara manual, karena koneksi client akan dihandle oleh Netty.

Yang hanya perlu kita buat adalah sebuah listener. Listener adalah sebuah objek sebagai target notifikasi ketika ada event yang terjadi. Netty akan secara otomatis memberi notifikasi kepada listener ketika ada sebuah event telah terjadi, seperti event ketika ada client yang terkoneksi, ketika ada pesan dari client yang diterima dan ketika client diskonek dari server.

Untuk membuat listener, kita hanya perlu membuat sebuah kelas turunan dari **`io.netty.channel.ChannelInboundHandlerAdapter`**, ada banyak metode yang bisa kita override di kelas tersebut, kita akan gunakan beberapa yang hanya kita perlukan saja. Sekarang ayo kita buat kelas ServerListener sebagai listener untuk aplikasi chat servernya.

```
package com.khannedy.chating;

import io.netty.channel.ChannelInboundHandlerAdapter;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ServerListener extends ChannelInboundHandlerAdapter {

}
```

## Listener ketika Client login

Listener yang akan kita buat pertama adalah ketika client Login terlebih dahulu. Kita akan membaca pesan ChatMessage dari client, jika ChatMessage tersebut atribut action=LOGIN, maka kita tambahkan client tersebut ke ChatDatabase.

Metode yang dipanggil oleh Netty ketika ada pesan masuk dari Client adalah metode **`channelRead(ChannelHandlerContext ctx, Object msg)`**

```
package com.khannedy.chating;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInboundHandlerAdapter;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ServerListener extends ChannelInboundHandlerAdapter {

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg)
            throws Exception {

        String json = msg.toString();
        ChatMessage chatMessage = ChatMessageConverter.convertFromJSON(json);

        if (chatMessage.getAction().equals(ChatAction.LOGIN)) {
            // chat action adalah login, tambahkan ke database
            ClientDatabase.login(ctx.channel());
            // beritahu client yang lain jika ada yang login
            ClientDatabase.chat(ctx.channel(), json);
        }

    }
}
```

## Listener ketika Client logout

Setelah logout, sekarang listener yang akan kita buat adalah ketika client mengirim pesan logout. Kita hanya perlu menambahkan else-if baru setelah listener login sebelumnya.

```
@Override
public void channelRead(ChannelHandlerContext ctx, Object msg)
        throws Exception {

    String json = msg.toString();
    ChatMessage chatMessage = ChatMessageConverter.convertFromJSON(json);

    if (chatMessage.getAction().equals(ChatAction.LOGIN)) {
        // chat action adalah login, tambahkan ke database
        ClientDatabase.login(ctx.channel());
        // beritahu client yang lain jika ada yang login
        ClientDatabase.chat(ctx.channel(), json);
    } else if (chatMessage.getAction().equals(ChatAction.LOGOUT)) {
        // chat action adalah logout, hapus dari database
        ClientDatabase.logout(ctx.channel());
       // beritahu client yang lain jika ada yang logout
        ClientDatabase.chat(ctx.channel(), json);
       // close koneksi client
        ctx.channel().close();
    }

}
```

## Listener ketika Client mengirim pesan

Listener yang terakhir adalah ketika client mengirim pesan.

```
    if (chatMessage.getAction().equals(ChatAction.LOGIN)) {
        // chat action adalah login, tambahkan ke database
        ClientDatabase.login(ctx.channel());
        // beritahu client yang lain jika ada yang login
        ClientDatabase.chat(ctx.channel(), json);
    } else if (chatMessage.getAction().equals(ChatAction.LOGOUT)) {
       // chat action adalah logout, hapus dari database
        ClientDatabase.logout(ctx.channel());
       // beritahu client yang lain jika ada yang logout
        ClientDatabase.chat(ctx.channel(), json);
       // close koneksi client
        ctx.channel().close();
    } else if (chatMessage.getAction().equals(ChatAction.CHAT)) {
        // kirim pesan ke client yang lain
        ClientDatabase.chat(ctx.channel(), json);
    }

}
```

Sekarang kita telah membuat listener untuk aplikasi chat yang akan kita buat. Selanjutnya kita perlu menambahkan Listener yang telah kita buat ke Netty agar Netty tau jika ada pesan yang masuk, maka dia akan mengirimkan notifikasi ke Listener yang telah kita buat.

## Membuat ServerInitializer

Untuk memberi tahu Netty bahwa kita telah membuat Listener, maka kita perlu membuat sebuah kelas turunan dari `ChannelInitializer<SocketChannel>`, disana kita bisa memberi tahu Netty bahwa kita telah membuat Listener, dan Netty akan otomatis mengirim notifikasi ke Listener yang telah kita buat.

```
package com.khannedy.chating;

import io.netty.channel.ChannelInitializer;
import io.netty.channel.socket.SocketChannel;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ServerInitializer extends ChannelInitializer<SocketChannel> {

    @Override
    protected void initChannel(SocketChannel socketChannel) throws Exception {
        // mengkonversi dari ByteBuf ke String JSON
        socketChannel.pipeline().addLast(new StringDecoder());
        socketChannel.pipeline().addLast(new StringEncoder());

        // menambahkan listener
        socketChannel.pipeline().addLast(new ServerListener());
    }
}
```

Pada kode diatas kita menambahkan **ServerListener** ke Netty, namun selain itu kita juga menambahkan **StringDecoder** dan **StringEncoder**. Untuk apa **StringEncode** dan **StringDecoder** tersebut?

Secara default, pesan yang dikirim ke client ke server ataupun dari server ke client merupakan sebuah objek **`io.netty.buffer.ByteBuf`**, yaitu ***array of byte***. Jadi kita perlu mengkonversi dari **String** (JSON) yang kita kirim menjadi **ByteBuf**. Karena jika konversi secara manual dari **String** ke **ByteBuf** cukup merepotkan, jadi lebih baik kita tambahkan **StringEncoder** dan **StringDecoder**, dimana **StringEncoder** akan secara otomatis mengkonversi **ByteBuf** menjadi **String**, dan **StringDecoder** akan secara otomatis mengkonversi dari **ByteBuf** menjadi **String**.

## Membuat ServerApp

Sekarang kita masuk ke bagian terakhir dari bab ini, yaitu membuat aplikasi server chat nya.

```
package com.khannedy.chating;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;

/**
 * @author Eko Khannedy
 */
public class ServerApp {
   public static void main(String[] args) throws InterruptedException {

        EventLoopGroup boosGroup = new NioEventLoopGroup();
        EventLoopGroup workerGroup = new NioEventLoopGroup();

        ServerBootstrap bootstrap = new ServerBootstrap()
                .group(boosGroup, workerGroup)
                .channel(NioServerSocketChannel.class)
                .childHandler(new ServerInitializer());

        bootstrap.bind(8000).sync().channel().closeFuture().sync();

        workerGroup.shutdownGracefully();
        boosGroup.shutdownGracefully();
    }
}
```

Kode diatas adalah cara menjalankan Server di Netty. Hal yang paling penting adalah port yang kita gunakan, kode diatas menggunakan port **8000**, dan juga **childHandler()** yang digunakan yaitu **ServerInitializer** yang telah kita buat.

Bab selanjutnya kita akan bahas cara membuat Client untuk aplikasi chat-nya.

# Aplikasi Chat Client

## yang digunakan oleh pengguna

Tahapan membuat Client untuk aplikasi chat, sama dengan tahapan membuat Server, yaitu kita perlu membuat Listener lalu memberi tahu Listener nya ke Netty menggunakan Initializer dan terakhir membuat Netty Client yang terkoneksi ke Netty Server (aplikasi chat server)

### Membuat ClientListener

Tugas listener untuk client sangat sederhana, yaitu menerima pesan dari server, dan menampilkannya. Disini kita hanya akan menggunakan terminal / command promt untuk menampilkan isi pesannya.

```
package com.khannedy.chating;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInboundHandlerAdapter;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ClientListener extends ChannelInboundHandlerAdapter {

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg)
                throws Exception {

        String json = msg.toString();
        ChatMessage chatMessage = ChatMessageConverter.convertFromJSON(json);
```

```
            if (chatMessage.getAction().equals(ChatAction.LOGIN)) {
                // ada client yang login
                String pesan = chatMessage.getSender() + " masuk chat room";
                System.out.println(pesan);

            } else if (chatMessage.getAction().equals(ChatAction.LOGOUT)) {
                // ada client yang logout
                String pesan = chatMessage.getSender() + " keluar chat room";
                System.out.println(pesan);

            } else if (chatMessage.getAction().equals(ChatAction.CHAT)) {
                // ada client mengirim pesan
                String pesan = chatMessage.getSender() + " : "
                        + chatMessage.getMessage();
                System.out.println(pesan);
            }

        }
    }
```

Berikut adalah contoh hasil ketika ada client yang login.

```
khannedy masuk chat room
```

Berikut adalah contoh hasil ketika ada client yang logout.

```
khannedy keluar chat room
```

Berikut adalah contoh hasil ketika ada client yang mengirim pesan.

```
khannedy : Hai apa kabar bro? dimana lo sekarang?
```

## Membuat ClientInitializer

Setelah membuat ClientListener, sekarang kita beritahu Netty melalui ClientlInitializer. Sama seperti pada Server, kita juga harus menambahkan StringEncoder dan StringDecoder agar tidak perlu manual mengkonversi String JSON ke ByteBuf.

```
package com.khannedy.chating;

import io.netty.channel.ChannelInitializer;
import io.netty.channel.socket.SocketChannel;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ClientInitializer extends ChannelInitializer<SocketChannel> {

    @Override
    protected void initChannel(SocketChannel socketChannel) throws Exception {
        // menambahkan string encoder dan decoder
        socketChannel.pipeline().addLast(new StringEncoder());
        socketChannel.pipeline().addLast(new StringDecoder());

        // menambahkan listener
        socketChannel.pipeline().addLast(new ClientListener());
    }
}
```

## Membuat ClientSender

Sebelum membuat client app, terlebih dahulu kita perlu membuat sebuah program untuk menerima input dari user, lalu mengirimkan ke server. Jika aplikasinya dalam bentuk desktop, mungkin ini adalah aplikasi GUI nya, tapi karena kita hanya akan menggunakan terminal / command promt, jadi tidak perlu rumit, cukup baca apa yang diketikkan oleh user, lalu convert jadi

ChatMessage, lalu konversi ke String JSON dan terakhir kirim ke Server.

```
package com.khannedy.chating;

import io.netty.channel.Channel;

import java.util.Date;
import java.util.Scanner;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ChatSender {

    /**
     * Scanner digunakan untuk menerima input dari user
     */
    private Scanner input = new Scanner(System.in);

    private String username;

    /**
     * Menjalankan chat sender
     *
     * @param server server channel
     */
    public void run(Channel server) {
        while (true) {
            // membaca pesan yang diketikkan user
            String message = input.nextLine();
            if (message.startsWith("login") && username == null) {
                // login user
                username = message.substring("login".length()).trim();
                sendLoginMessage(server);
            } else if (message.equals("logout")) {
                // logout user
                sendLogoutMessage(server);
                // selesai
                break;
            } else {
                // chat user
                sendChatMessage(server, message);
            }
        }
    }
}
```

Ada 3 metode yang belum kita implementasikan, yaitu sendLoginMessage(), sendLogoutMessage() dan sendChatMessage().

sendLoginMessage() digunakan untuk mengirim pesan login ke server

```
/**
 * Mengirim pesan login ke server
 *
 * @param server server channel
 */
private void sendLoginMessage(Channel server) {
    // buat objek chat message
    ChatMessage chatMessage = new ChatMessage();
    chatMessage.setAction(ChatAction.LOGIN);
    chatMessage.setDatetime(new Date());
    chatMessage.setSender(username);

    // konver ke json dan kirim ke server
    String json = ChatMessageConverter.convertToJSON(chatMessage);
    server.writeAndFlush(json);
}
```

sendLogoutMessage() digunakan untuk mengirim pesan logout ke Server

```
/**
 * Mengirim pesan logout ke server
 *
 * @param server server channel
 */
private void sendLogoutMessage(Channel server) {
    // buat objek chat message
    ChatMessage chatMessage = new ChatMessage();
    chatMessage.setAction(ChatAction.LOGOUT);
    chatMessage.setDatetime(new Date());
    chatMessage.setSender(username);

    // konver ke json dan kirim ke server
    String json = ChatMessageConverter.convertToJSON(chatMessage);
    server.writeAndFlush(json);
}
```

Dan yang terakhir adalah sendChatMessage() digunakan untuk mengirim pesan chat ke server.

```
/**
 * Mengirim pesan chat ke seluruh client
 *
 * @param server  server channel
 * @param message pesan chat
 */
private void sendChatMessage(Channel server, String message) {
    // buat objek chat message
    ChatMessage chatMessage = new ChatMessage();
    chatMessage.setAction(ChatAction.CHAT);
    chatMessage.setDatetime(new Date());
    chatMessage.setSender(username);
    chatMessage.setMessage(message); // jangan lupa pesannya

    // konver ke json dan kirim ke server
    String json = ChatMessageConverter.convertToJSON(chatMessage);
    server.writeAndFlush(json);
}
```

Walaupun kode ChatSender lumayan panjang, tapi sebenarnya sangat sederhana, yaitu membuat objek ChatMessage lalu mengkonversinya menjadi String JSON setelah itu mengirim ke server. Jadi jangan tertipu dengan panjangnya kode, padahal logic nya sangat sederhana :D

## Membuat ClientApp

Bagian terakhir dari bab ini adalah membuat ClientApp, dimana kita aman menggunakan Netty juga sebagai client nya, jadi disini yang kita buat adalah Netty Client yang akan terkoneksi ke Netty Server.

```
package com.khannedy.chating;

import io.netty.bootstrap.Bootstrap;
import io.netty.channel.Channel;
import io.netty.channel.EventLoopGroup;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioSocketChannel;

/**
 * @author Eko Khannedy
 * @since 11/27/14
 */
public class ClientApp {

    public static void main(String[] args) throws InterruptedException {
        EventLoopGroup group = new NioEventLoopGroup();

        Bootstrap bootstrap = new Bootstrap()
                .group(group)
                .channel(NioSocketChannel.class)
                .handler(new ClientInitializer());

        // konek ke server berlokasi di localhost dengan port 8000
        Channel server = bootstrap.connect("localhost", 8000).sync().channel();

        // jalankan ChatSender
        ChatSender chatSender = new ChatSender();
        chatSender.run(server);

        group.shutdownGracefully();
    }

}
```

Pada kode diatas, saya menggunakan localhost sebagai lokasi server nya, karena memang akan dijalankan di komputer yang sama antara client dan server. Jika Anda menggunakan komputer yang berbeda, ubah localhost menjadi **IP Address** komputer server. Untuk port yang digunakan, pastikan sama dengan port yang digunakan server, disini saya menggunakan port **8000**. Sekarang kita telah selesai membuat aplikasi chat, saatnya mencobanya! #yes

# Menjalankan Aplikasi

## yuk coba aplikasi chat nya

Sekarang saatnya kita mencoba aplikasi chatnya, pertama sebelum mencoba chating, terlebih dahulu jalankan aplikasi ServerApp nya.

> Jika anda menggunakan IDE (seperti NetBeans, Eclipse dan IntelliJ IDEA, sangat mudah untuk menjalankan ServerApp, cukup klik kanan file ServerApp.java nya lalu pilih **Run File**.

### Menjalankan ServerApp

Untuk menjalankan aplikasi ServerApp nya, kita bisa menggunakan perintah berikut di terminal.

```
mvn compile

mvn exec:java -Dexec.mainClass="com.khannedy.chating.ServerApp"
```

Ubah lokasi package kelas ServerApp menjadi package yang Anda buat, disini saya menggunakan package "**com.khannedy.chatting**"

### Menjalankan ClientApp

Untuk menjalankan ClientApp sama saja seperti menjalankan ServerApp, hanya saja tinggal kita ubah class nya menjadi ClientApp.

Berikut kode untuk menjalankan ClientApp

```
mvn compile

mvn exec:java -Dexec.mainClass="com.khannedy.chating.ClientApp"
```

## Login

Sebelum melakukan chating pastikan Kita login terlebih dahulu, caranya cukup gunakan perintah

```
login Nama User
```

## Mengirim Pesan

Setelah login, kita tinggal mengetikkan pesan apa saja untuk dikirim ke server, misal

```
Hai bro? ada yang online kah?
```

## Logout

Terakhir, setelah selesai chating, jika ingin keluar, cukup gunakan perintah

```
logout
```

## Akhirnya

Sekarang Anda telah selesai membuat aplikasi chat, berikut saya beri contoh percakapan chat menggunakan 3 user.

User eko

```
login eko
kurniawan masuk chat room
khannedy masuk chat room
khannedy : ada yang lain online?
kurniawan : gw online bro :D
gw juga online nih :D
khannedy : ya udah, gw cuma mau nyapa doank kok
khannedy : hahahhaa
khannedy : gw logout lagi ya :D
khannedy keluar chat room
dasar dodol, hahahahha
gw juga mau off dulu ahhhh
logout
```

User kurniawan

```
login kurniawan
khannedy masuk chat room
khannedy : ada yang lain online?
gw online bro :D
eko : gw juga online nih :D
khannedy : ya udah, gw cuma mau nyapa doank kok
khannedy : hahahhaa
khannedy : gw logout lagi ya :D
khannedy keluar chat room
eko : dasar dodol, hahahahha
eko : gw juga mau off dulu ahhhh
eko keluar chat room
ya udah gw logout juga
logout
```

User khannedy

```
login khannedy
ada yang lain online?
kurniawan : gw online bro :D
eko : gw juga online nih :D
ya udah, gw cuma mau nyapa doank kok
hahahhaa
gw logout lagi ya :D
logout
```

# Saya Nyerah!

## masa gitu aja gak bisa :P

Jika Anda merasa ada masalah ketika mengikuti buku ini. Kok udah bener, tapi error. Perasaan udah copy-paste, tapi tetep error. Oke, tenang saja :D

Jika anda menyerah untuk praktek sendiri, atau mungkin tujuan anda untuk tugas kuliah, atau untuk bikinin tugas gebetan. Oke, saya akan permudah jalan anda, dengen memberikan source code nya secara lengkap.

Anda bisa download source code buku ini secara lengkap di GitHub saya disini :

https://github.com/khannedy/aplikasi-chating

Gak ngerti cara pake Git? Oke, akan saya permudah lagi, maklum saya kan baik hati dan tidak sombong :D Silahkan download ZIP nya disini :

https://github.com/khannedy/aplikasi-chating/archive/master.zip

# What's Next?

## hal yang perlu Anda lanjutkan

Aplikasi chat yang sudah kita buat masih belum sempurna, masih banyak yang harus diperbaiki dan ditingkatkan, namun tidak akan saya bahas di buku ini, karena bukunya bisa kepanjangan jika dibahas. Jadi saya sarankan Anda saja yang meneruskan aplikasi chatnya :D

Berikut adalah hal yang bisa ditingkatkan di aplikasi chat yang telah kita buat.

- Tidak ada **validasi username** di Server, dengan demikian user dapat menggunakan username yang sama dengan user lain.
- Di ClientSender, user **bisa mengirim pesan walaupun belum login**, silahkan perbaiki.
- Di ClientListener, **informasi waktu pengiriman** pesan tidak ditampilkan, silahkan tampilkan di chat.
- Silahkan buat versi **Apliksi GUI** untuk client nya, sehingga lebih menarik dibandingkan chating menggunakan terminal / command promt.
- Ada lagi? :D

# Penulis

## Eko Kurniawan Khannedy

Penulis adalah seorang coder yang sudah coding dibangku SMA (walaupun cuma if-else pake JavaScript). Penulis tertarik ke dunia Java semasa kuliah (2007). Saat ini selain Java, penulis juga terbiasa menggunakan Scala dan NodeJS.

Penulis sudah terbiasa membuat ebook dan screencast seputar programming. Penulis juga sering menjadi pembicara di **seminar**, **workshop** dan **pelatihan** seputar teknologi Java.

Saat ini penulis sibuk mengembangkan startup-nya **@MusterLabs** yang bergerak di bidang *Artifical Intelligence* dan *Machine Learning*. Namun disela-sela kesibukannya, penulis juga selalu menyempatkan diri dengan berjualan **@MieAyamMbot** dan **@sate_mbot** :)

Penulis dapat dihubungi melalui :

- Twitter : @khannedy
- Facebook : fb.com/khannedy
- WhatsApp : 0812-9549-7313 (Telpon akan di REJECT ajah :D)
- Email : echo.khannedy@gmail.com