

2010

Membangun Aplikasi Client-Server Menggunakan Java



Eko Kurnaiwan Khannedy

eecchoo.wordpress.com

3/17/2010

Peringatan

Buku ini bukan untuk orang yang baru belajar Java, apa yang cuma tau kalo Java = Jawa, Hahah. Tapi buku ini dibuat untuk orang yang tau apa itu Java dan mengerti pemrograman berorientasi objek dengan Java dan juga mengerti sedikit banyaknya tentang membuat Swing atau lebih dikenal dengan Java GUI.

Jika Anda baru mengenal Java, saya sarankan untuk menunda dulu membaca buku ini, dan terlebih dahulu membaca buku tentang Pengenalan Java dan juga saya anjurkan untuk membaca buku tentang Java Swing (Java GUI).

Selamat Datang

Selamat datang di buku “Membangun Aplikasi Client Server dengan Java”. Dimana dalam buku ini saya akan membahas tentang cara membangun aplikasi Client Server dengan Java memanfaatkan fasilitas yang dimiliki oleh Java yaitu RMI.

Saat ini jarang sekali aplikasi yang dibangun standalone, walaupun ada biasanya program tersebut bersifat general seperti OpenOffice, GIMP, Photoshop. Kebanyakan aplikasi saat ini yang dibuat untuk mengelola Database, biasanya dibangun menggunakan konsep Client Server.

Misal saja aplikasi perpustakaan akan lebih baik jika dibangun menggunakan konsep Client Server, kenapa? Jika kita menggunakan konsep Client Server, maka seluruh aktifitas manipulasi data ke database akan terpusat di Server, sehingga Client tidak akan tau menahu tentang urusan Database, Client hanya merequest dan Server lah yang melakukan. Hal ini sangat baik karena Client tidak akan bekerja keras karena semuanya dikerjakan oleh Server, selain itu tingkat keamanan Sistem akan lebih terkontrol karena Client tidak akan langsung berhubungan dengan Database, melainkan harus lewat Server.

Saya sadar kalau buku ini bukanlah buku yang sempurna. Buku ini hanyalah buku yang saya buat untuk membantu Anda dalam teknik membangun aplikasi client server menggunakan Java. Oleh sebab itu kritik dan saran akan sangat saya harapkan dari Anda, sehingga saya bisa melakukan perbaikan terus menerus agar karya-karya saya selanjutnya bisa lebih baik dari ini.

Penulis

Bandung, Maret 2010

Daftar Isi

Peringatan	1
Selamat Datang	2
Daftar Isi	3
Daftar Gambar.....	5
Daftar Kode	7
Apa Itu RMI?	8
Syarat object bisa diakses lewat RMI.....	9
Class harus implements Interface.....	9
Interface harus extends java.rmi.Remote	9
Class harus extends java.rmi.server.UnicastRemoteObject.....	10
Class harus memiliki konstruktor yang throw java.rmi.RemoteException	10
Seluruh metode Interface harus throw java.rmi.RemoteException	10
Membuat Server RMI	12
Menyimpan Object ke Server	12
Menghapus Object dari Server	13
Membuat Client RMI	14
Mengakses Object yang ada di Server	14
Menjalankan RMI	15
Menjalankan Server.....	15
Menjalankan Client.....	16
Aplikasi SayHello Berbasih Client-Server	19

Membuat Interface SayHello	19
Membuat Class SayHelloServer	19
Cara membuat project Client Server di NetBeans IDE	20
Membuat Server SayHello	37
Membuat Client SayHello	38
Menjalankan Aplikasi SayHello	47
Lampiran	52
Instalasi JDK.....	52
Tentang Saya	56

Daftar Gambar

Gambar 1 Contoh Implementasi RMI	8
Gambar 2 Membuka NetBeans IDE	21
Gambar 3 NetBeans IDE 6.5	22
Gambar 4 Dialog New Project	22
Gambar 5 Java Class Library	24
Gambar 6 New Java Class Library	24
Gambar 7 Project Manager	25
Gambar 8 Java Application	25
Gambar 9 New Java Application	26
Gambar 10 Tampilan Project SayHello Server	27
Gambar 11 New Java Application	28
Gambar 12 Menambah Project	29
Gambar 13 Pilih Project say-hello-api	29
Gambar 14 Bagian-Bagian Project	30
Gambar 15 New Package	31
Gambar 16 New File	31
Gambar 17 New Java Package	32
Gambar 18 Package Baru	32
Gambar 19 New Interface	33
Gambar 20 New File	33
Gambar 21 New Java Interface	34
Gambar 22 Interface SayHello	34
Gambar 23 New Class	35
Gambar 24 New File	36
Gambar 25 New Java Class	36
Gambar 26 Class SayHelloServer	37
Gambar 27 New File	39
Gambar 28 New JFrame Form	39
Gambar 29 GUI Builder pada NetBeans IDE	40
Gambar 30 Pallette	41
Gambar 31 Properties	42
Gambar 32 Tampilan Aplikasi SayHello	42

Gambar 33 Nama-Nama Variabel Form Client SayHello.....	43
Gambar 34 Mengubah Nama Variabel	44
Gambar 35 Rename.....	44
Gambar 36 Inspector	44
Gambar 37 Button Source	45
Gambar 38 Konstruktor Form Client	45
Gambar 39 Menambah ActionListener	46
Gambar 40 metode buttonSayHelloActionPerformed	46
Gambar 41 Isi metode buttonSayHelloActionPerformed	47
Gambar 42 Error Client ketika dijalankan	48
Gambar 43 Menjalankan Server	48
Gambar 44 Output Trace Server	48
Gambar 45 Aplikasi Client	49
Gambar 46 Response dari Server	49
Gambar 47 Trace Respon Server	50
Gambar 48 Mematikan Server	50
Gambar 49 Error pada Client ketika Server mati	50

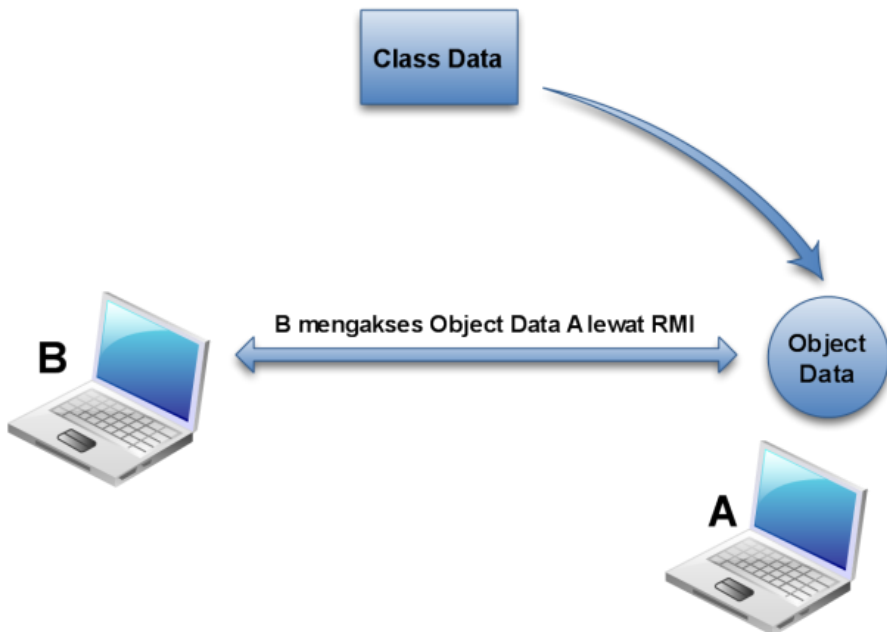
Daftar Kode

Kode 1 Interface DataInterface	9
Kode 2 Class Data	9
Kode 3 Interface DataInterface	10
Kode 4 Class Data	10
Kode 5 Class Data	10
Kode 6 Interface DataInterface	11
Kode 7 Class Data	11
Kode 8 Membuat Registry	12
Kode 9 Menyimpan Object ke Server	12
Kode 10 Metode rebind()	13
Kode 11 Menghapus object dari Server	13
Kode 12 Membuat Client RMI	14
Kode 13 Mengakses Data di Server	14
Kode 14 Class Server	16
Kode 15 Class Client	17
Kode 16 Output Client	17
Kode 17 Output Server	17
Kode 18 Interface SayHello	19
Kode 19 Class SayHelloServer	20
Kode 20 Class Main Server	38
Kode 21 Membuat Registry Client	45
Kode 22 Isi metode buttonSayHelloActionPerformed	46
Kode 23 Class Main Client	47

Apa Itu RMI?

RMI alias Remote Method Invocation merupakan fasilitas yang dimiliki Java untuk pemanggilan Object dari sisi yang berbeda, anggap saja sisi yang menyediakan Object adalah server dan sisi memanggil Object adalah Client.

Dimisalkan seperti ini, A adalah Server dan B adalah Client. A membuat object dari class Data dengan nama object data. Kemudian B ingin membuat Object dari class Data, namun B ingin Object nya itu Object data milik A. Nah dengan menggunakan RMI, B bisa mengakses Object data milik A, walaupun A dan B tidak terletak pada satu komputer.



Gambar 1 Contoh Implementasi RMI

Jadi dengan kata lain, RMI merupakan jembatan penghubung antara satu Aplikasi dengan Aplikasi lainnya, saya tidak menyebutkan antara satu Komputer dengan Komputer lain karena pada kenyataannya RMI bisa diimplementasikan dalam satu komputer, contohnya dalam buku ini saya mempraktekan RMI dalam satu komputer.

Syarat object bisa diakses lewat RMI

Agar sebuah object dapat diakses lewat RMI, kita tidak bisa menggunakan object dari class sembarangan. Ada syarat tertentu agar sebuah object bisa diakses lewat RMI. Untuk syarat-syaratnya antara lain :

Class harus implements Interface

Maksud implements Interface adalah, object yang dibuat dari class A (misalkan), harus merupakan turunan dari sebuah interface, misal contohnya kita memiliki class Data, agar bisa diakses lewat RMI class Data tersebut harus merupakan turunan dari sebuah Interface, misal saja kita buat interface DataInterface :

```
package com.echo.clientserver.apaiturmi;  
  
public interface DataInterface {  
  
}
```

Kode 1 Interface DataInterface

```
package com.echo.clientserver.apaiturmi;  
  
public class Data implements DataInterface {  
  
}
```

Kode 2 Class Data

Interface harus extends java.rmi.Remote

Selain Class harus turunan Interface, Interface tersebut harus extends java.rmi.Remote, sehingga menjadi seperti ini :

```
package com.echo.clientserver.apaiturmi;  
  
import java.rmi.Remote;  
  
public interface DataInterface extends Remote {  
  
}
```

Kode 3 Interface DataInterface

Class harus extends java.rmi.server.UnicastRemoteObject

Selain Interface yang harus extends java.rmi.Remote, class yang mengimplementasikan Interface tersebut harus extends java.rmi.server.UnicastRemoteObject, sehingga seperti ini :

```
package com.echo.clientserver.apaiturmi;

import java.rmi.server.UnicastRemoteObject;

public class Data extends UnicastRemoteObject implements DataInterface {

}
```

Kode 4 Class Data

Class harus memiliki konstruktor yang throw java.rmi.RemoteException

Selain class object yang akan diakses lewat RMI harus extends java.rmi.server.UnicastRemoteObject, class tersebut juga harus memiliki konstruktor yang throw java.rmi.RemoteException, sehingga seperti ini :

```
package com.echo.clientserver.apaiturmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Data extends UnicastRemoteObject implements DataInterface {

    public Data() throws RemoteException {

    }

}
```

Kode 5 Class Data

Seluruh metode Interface harus throw java.rmi.RemoteException

Pada contoh diatas kita belum membuat satu metode pun di Interface, namun jika kita akan membuat Interface yang nantinya objectnya akan di

akses lewat RMI, seluruh metode yang dimiliki Interface tersebut harus throw `java.rmi.RemoteException`, sehingga seperti ini :

```
package com.echo.clientserver.apaiturmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface DataInterface extends Remote {

    public void metodeSatu() throws RemoteException;

    public void metodeDua() throws RemoteException;
}
```

Kode 6 Interface DataInterface

```
package com.echo.clientserver.apaiturmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Data extends UnicastRemoteObject implements DataInterface {

    public Data() throws RemoteException {
    }

    public void metodeSatu() throws RemoteException {
        System.out.println("Metode Satu Dijalankan");
    }

    public void metodeDua() throws RemoteException {
        System.out.println("Metode Dua Dijalankan");
    }
}
```

Kode 7 Class Data

Syarat-syarat diatas lah yang harus dimiliki oleh class dan interface yang objectnya nanti bisa diakses lewat RMI. Agak sedikit rumit memang, namun hal ini memang wajib jika kita menggunakan RMI.

Membuat Server RMI

Setelah tadi kita bahas tentang syarat-syarat object yang bisa diakses lewat RMI, sekarang kita bahas tentang bagaimana membuat Server menggunakan RMI. Server ini nantinya digunakan sebagai tempat untuk melakukan proses sharing object, sehingga Client bisa mengakses object RMI ke Server tersebut.

Dalam RMI, Server direpresentasikan sebagai interface `java.rmi.registry.Registry`, dan karena itu merupakan Interface, oleh karena itu kita bisa membuat object `java.rmi.registry.Registry` dengan menggunakan instansiasi "new". Namun untuk membuat object registry kita bisa menggunakan metode `createRegistry(int port)` milik class `java.rmi.registry.LocateRegistry` :

```
Registry registry = LocateRegistry.createRegistry(1099);
```

Kode 8 Membuat Registry

Dalam kode diatas saya membuat Server (Registry) RMI di port 1099, ini adalah port yang biasa digunakan oleh RMI, namun kita juga bisa merubahnya sesuai dengan yang kita mau, namun hati-hati dalam penggunaan nomor port, karena jika nomor port yang kita buat untuk Server RMI telah terpakai, maka akan terjadi proses Error, misal biasanya MySQL menggunakan port 3306, Oracle menggunakan port 1521, jadi saya sarankan gunakan saja nomor port 1099 yang menjadi port default RMI.

Menyimpan Object ke Server

Setelah tadi kita membuat Server, terlebih dahulu kita harus menyimpan Object ke Server agar bisa diakses oleh Client, dan tentunya Object yang akan kita share adalah Object dari class yang tadi kita buat, dan untuk menyimpan Object ke Server kita bisa menggunakan perintah :

```
Data data = new Data();  
registry.bind("data", data);
```

Kode 9 Menyimpan Object ke Server

Metode bind() tersebut memiliki dua parameter, pertama adalah String dan yang kedua adalah Object. String yang dimaksud adalah nama Object yang disimpan, nama object tersebut bisa bebas sesuai dengan yang kita inginkan, namun harus diingat untuk mengakses object tersebut di client, kita harus menggunakan nama tersebut, jadi nama tersebut harus benar-benar bisa diingat, karena andai berbeda satu huruf, maka client tidak bisa mendapatkan object yang kita maksud. Dan untuk parameter yang kedua adalah object yang akan kita share. Untuk contoh diatas adalah saya melakukan proses penyimpanan data object class Data dengan nama "data".

Yang harus diingat dalam penggunaan metode bind() adalah, andai kita menyimpan object dengan nama yang telah ada di Server, maka akan terjadi error, misal kita menyimpan object dengan nama "data" setelah itu kita juga menyimpan object yang baru lagi dengan maksud akan memperbaharui object data sebelumnya dengan nama yang sama. Maka hal itu akan menyebabkan error. Sehingga jika kita akan menyimpan object yang suatu waktu bisa berubah saya sarankan menggunakan metode rebind() :

```
Data data = new Data();  
registry.rebind("data", data);
```

Kode 10 Metode rebind()

Cara kerja metode rebind() adalah pertama Server akan mendeteksi apakah object dengan nama yang sama telah ada, jika belum ada maka object yang baru akan disimpan, namun jika object dengan nama yang sama ada, maka object yang lama akan dihapus dan digantikan oleh object yang baru.

Menghapus Object dari Server

Setelah tadi kita membahas tentang cara menyimpan object ke Server, pastinya kita juga harus tau bagaimana cara menghapus object tersebut dari server. Untuk melakukan proses tersebut, caranya cukup sederhana yaitu tinggal menggunakan metode unbind(String nama) milik java.rmi.registry.Registry :

```
registry.unbind("data");
```

Kode 11 Menghapus object dari Server

Tapi harus diingat, kita hanya bisa menghapus data yang memang ada, misal pada kode diatas saya menghapus object dengan nama “data”, andai saja object dengan nama data tidak ada, maka akan terjadi error pada proses penghapusan tersebut, jadi harus dipastikan bahwa object dengan nama “data” memang ada di Server.

Membuat Client RMI

Setelah tadi kita membuat Server, sekarang saatnya kita membuat Client untuk mengakses object yang ada di Server. Berbeda dengan membuat Server, untuk membuat Client, kita menggunakan class `java.rmi.Naming` :

```
java.rmi.Naming
```

Kode 12 Membuat Client RMI

Yang jadi permasalahan adalah class `Naming` merupakan class `Utilities`, artinya kita tidak bisa membuat object `Naming` dengan instansiasi “new”, sehingga kita cukup mengetikkan `java.rmi.Naming` untuk membuat Client.

Mengakses Object yang ada di Server

Setelah tadi kita membuat Client, sekarang kita bahas tentang cara mengakses object yang ada di Server, contohnya pada kode sebelumnya kita telah menyimpan object class `Data` di Server dan sekarang kita akan mengaksesnya.

Untuk mengakses data yang ada di server, kita harus membuat object Interfacenya, bukan classnya. Misal karena kita tadi menyimpan object class `Data` di server, maka di client kita mambut object interface `DataInterface` yang merupakan Interface yang diimplementasikan oleh class `Data`. Dan untuk membuat object Interface tersebut kita menggunakan :

```
DataInterface data =  
(DataInterface) Naming.lookup("rmi://localhost:1099/data");
```

Kode 13 Mengakses Data di Server

Untuk mengakses data tersebut kita bisa menggunakan metode `lookup` milik `Naming` dengan parameter url yang ada di server tadi, karena kita

sebelumnya memberinama object tersebut dengan nama "data" maka di Client-pun kita mengaksesnya dengan nama "data". Dan yang perlu diingat adalah jika object dengan nama "data" tidak ada di Server, maka proses pengaksesan object tersebut akan error.

Untuk ketentuan url pengaksesan object di server adalah sebagai berikut :

"rmi://" + host + ":" + port + "/" + nama_object

Dimana host adalah tempat server, bisa nama host atau juga bisa IP Address, untuk port adalah port yang digunakan oleh Server dan untuk nama_object adalah nama object yang ada di Server.

Menjalankan RMI

Setelah tadi kita membuat Object yang akan diakses lewat RMI, lalu membuat Server dan terakhir membuat Client, sekarang kita akan coba menjalankan kode diatas.

Menjalankan Server

Untuk menjalankan Server, kita cukup menggabungkan kode-kode diatas yang tadi dalam bagian Membuat Server RMI kedalam class yang memiliki metode utama :


```
package com.echo.clientserver.apaiturmi;

import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Server {

    public static void main(String[] args) throws RemoteException,
NotBoundException {
        Registry registry = LocateRegistry.createRegistry(1099);

        Data data = new Data();
        registry.rebind("data", data);

        System.out.println("Server Berhasil Berjalan");
    }
}
```

Kode 14 Class Server

Jika dijalankan maka akan terlihat trace dengan tulisan “Server Berhasil Berjalan”. Namun jika gagal biasanya kegagalan terjadi pada port.

Menjalankan Client

Untuk menjalankan Client, kita juga hanya perlu menggabungkan seluruh kode pada bagian membuat Client RMI dalam satu class yang memiliki metode utama :

```

package com.echo.clientserver.apaiturmi;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class Client {

    public static void main(String[] args) throws RemoteException,
    NotBoundException, MalformedURLException {
        DataInterface data =
        (DataInterface) Naming.lookup("rmi://localhost:1099/data");
        System.out.println("Client Berhasil Terkoneksi Ke Server");

        // Jalankan Aksi Object
        data.metodeSatu();
        data.metodeDua();

        System.out.println("Client Telah Selesai");
    }
}

```

Kode 15 Class Client

Perlu diingat, sebelum menjalankan Client, pastikan bahwa Server telah berjalan. Jika Server telah berjalan coba jalankan Client (kode diatas). Maka dalam trace output program akan terlihat seperti ini :

```

Client Berhasil Terkoneksi Ke Server
Client Telah Selesai

```

Kode 16 Output Client

Setelah itu liat trace output pada Server, maka sekarang akan terlihat seperti ini :

```

Server Berhasil Berjalan
Metode Satu Dijalankan
Metode Dua Dijalankan

```

Kode 17 Output Server

Dengan demikian kita telah berhasil membuat aplikasi sederhana berbasis Client Server, dimana Client memerintah sesuatu dan Server menjalankannya. Pada bab-bab selanjutnya kita akan bahas lagi tentang aplikasi yang lebih kompleks dari ini.

Aplikasi SayHello Berbasis Client-Server

Sekarang kita akan coba membuat aplikasi Say Hello berbasis client server. Dalam aplikasi ini Client akan menginputkan nama setelah itu Server akan membalas dengan mengatakan “Hello” + nama. Aplikasi ini akan kita buat berbasis GUI agar lebih bagus.

Membuat Interface SayHello

Pertama kali dalam membuat aplikasi client server adalah kita membuat interface untuk class dan object yang akan kita akses lewat Server. Misalnya sekarang kita berinama interfacenya dengan nama SayHello, karena dalam kasus ini kita hanya akan melakukan proses pengucapan “Hello” yang dilakukan oleh Server, maka kita hanya perlu membuat satu metode dalam interface SayHello yang mengembalikan (return) nilai String :

```
package com.echo.clientserver.sayhello;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface SayHello extends Remote {

    public String sayHello(String nama) throws RemoteException;
}
```

Kode 18 Interface SayHello

Pada metode sayHello() diatas kita menambahkan parameter String nama yang akan kita gunakan untuk melakukan respon String “Hello” + nama.\

Membuat Class SayHelloServer

Saya sengaja memberi nama class SayHello dengan akhiran Server yang menandakan bahwa class ini adanya di pihak server, artinya pihak Client tidak akan tau menau tentang class ini. Untuk membuat class SayHelloServer caranya hampir mirip seperti membuat class Data pada bab sebelumnya

yaitu pertama implements SayHello lalu extends UnicastRemoteObject dan setelah itu buat konstruktor dengan throw RemoteException :

```
package com.echo.clientserver.sayhello.server;

import com.echo.clientserver.sayhello.SayHello;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class SayHelloServer extends UnicastRemoteObject implements SayHello {

    public SayHelloServer() throws RemoteException {
    }

    public String sayHello(String nama) throws RemoteException {
        System.out.println("Client Dengan Nama " + nama + " Melakukan Request");
        return "Hello " + nama;
    }
}
```

Kode 19 Class SayHelloServer

Dari kode diatas kita telah memberi isi pada metode sayHello dengan pertama jika client memanggil metode tersebut, maka akan ada tulisan "Client Dengan Nama xxxx Melakukan Request", setelah itu mengembalikan (return) String "Hello xxxxx" yang nantinya akan didapatkan oleh Client sebagai respon dari Server.

Cara membuat project Client Server di NetBeans IDE

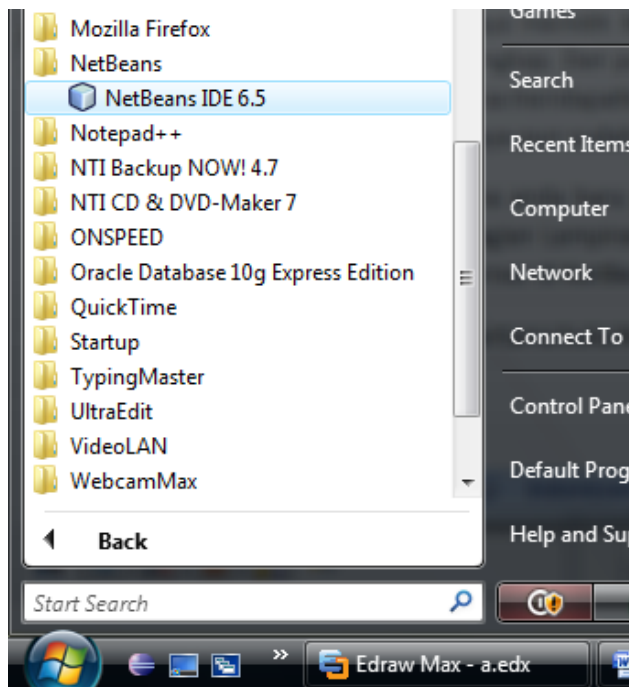
Bagi yang baru pertama kali membuat project client server mungkin akan sedikit bingung dengan cara membaut project client server. Kenapa? Nyatanya memang membaut project client server dan project apliaksi standalone memang sangat berbeda.

Dalam buku ini saya akan bahas tentang membaut project client server menggunakan NetBeans IDE, saya memilih NetBeans IDE karena fasilitas yang diberikan oleh NetBeans IDE memang sangat lengkap. Dan pada buku

ini saya menggunakan NetBeans IDE 6.5. Anda tak perlu khawatir dimana bisa mendapatkan NetBeans IDE, karena emang perangkat lunak ini Free dan OpenSource sehingga saya-pun sudah menyediakan NetBeans IDE dalam CD buku ini.

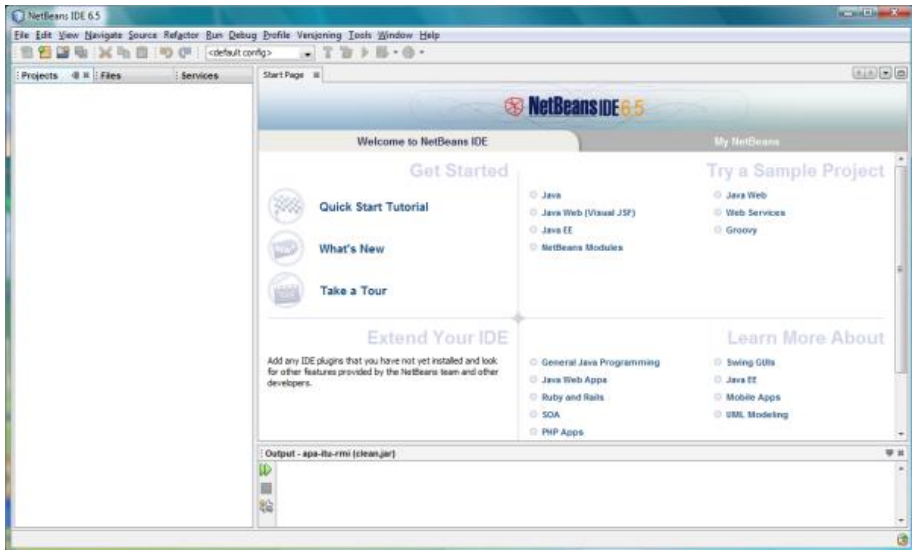
Jika anda baru pertama kali menggunakan NetBeans IDE, anda bisa lihat proses Instalasinya pada bagian Lampiran. Dan sekarang saya akan beri sedikit step by step teknik membuat aplikasi client server di NetBeans IDE.

Pertama buka NetBeans IDE lewat menu **Windows -> All Program -> NetBeans -> NetBeans IDE 6.5**



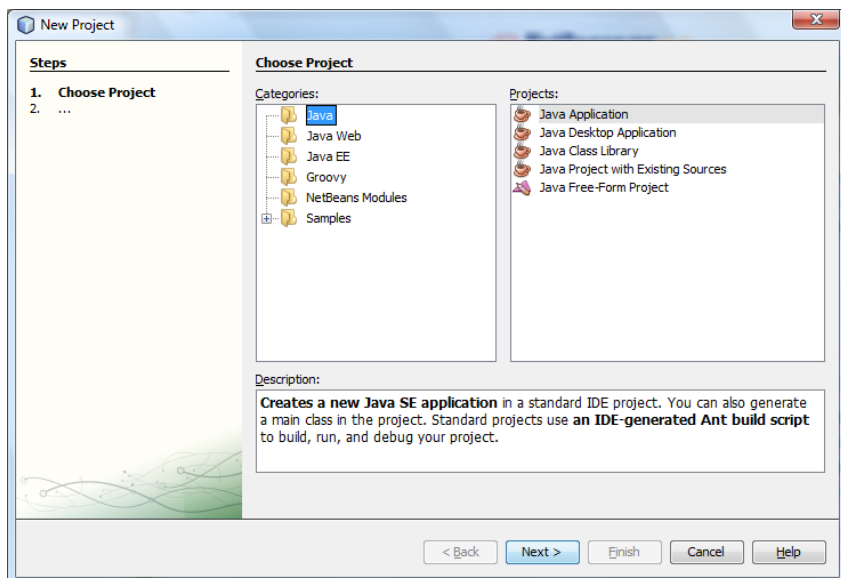
Gambar 2 Membuka NetBeans IDE

Aplikasi NetBeans IDE terlihat seperti pada gambar dibawah ini :



Gambar 3 NetBeans IDE 6.5

Setelah itu untuk membuat project java, kita bisa membuatnya lewat menu **File -> New Project** sehingga akan muncul dialog seperti dibawah ini :



Gambar 4 Dialog New Project

Dalam membuat project client server, kita akan membuat tiga project, *“Kenapa tiga project? Tidak dua project? Bukannya cukup dua project, project pertama untuk Client dan project kedua untuk Server”* Pertanyaan seperti itu sering saya lihat di forum-forum. Jawabannya karena kita memang butuh tiga project.

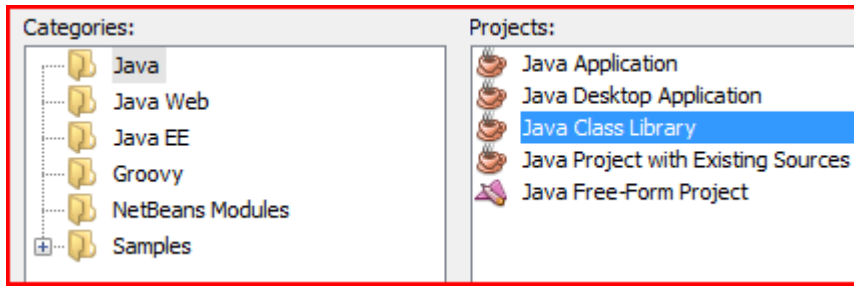
Project pertama berisikan interface misal pada project ini adalah interface SayHello. Dalam project ini berisikan seluruh class yang bersifat Interface atau dengan kata lain dibutuhkan oleh semua project baik itu Client maupun Server. Pada contoh sebelumnya kita bisa liat bahwa DataInterface dibutuhkan di sisi Client dan Server, di sisi Server DataInterface digunakan sebagai interface yang diimplementasi oleh class Data dan di sisi Client DataInterface digunakan untuk memanggil (lookup) data dari server lewat RMI.

Project kedua merupakan project Server, dalam project ini berisikan seluruh class Server yang artinya tidak boleh diketahui oleh Client dan tidak ada sangkut pautnya dengan Client. Selain itu di project ini kita membuat class implementasi dari interface pada project pertama, pada bagian ini class SayHelloServer berada dalam project ini.

Project ketiga merupakan project Client, dalam project ini berisikan seluruh class Client. Dan dalam project inilah kita mengakses Server lewat RMI.

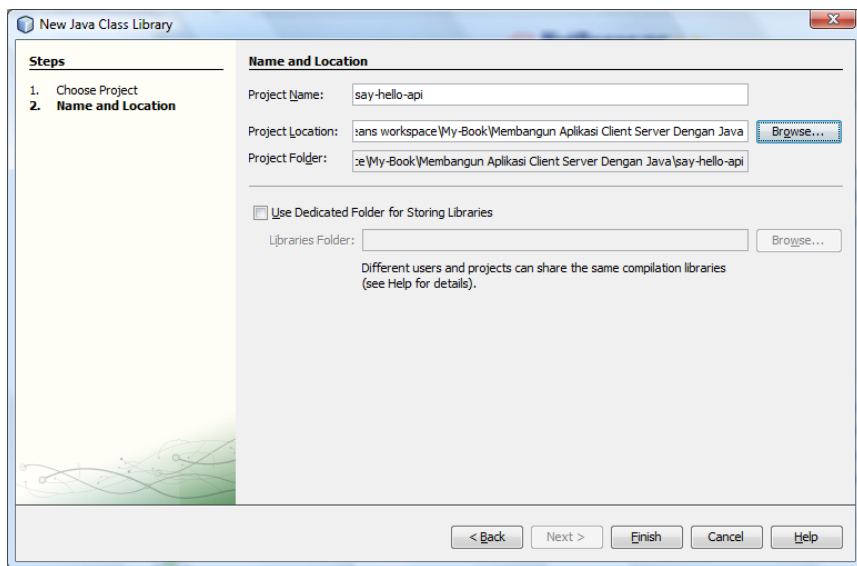
Kembali ke pokok masalah, sekarang pertama kita akan membuat project pertama dengan nama say-hello-api¹, karena pada project ini kita bukan membuat Aplikasi, maka pilih Categories Java dan pilih Projects Java Class Library :

¹ API (Application Programming Interface) atau lebih dikenal dengan kumpulan Interface



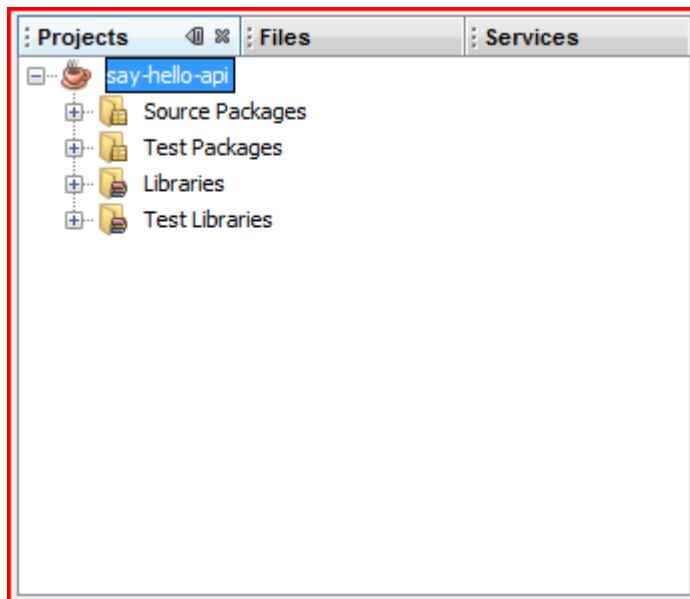
Gambar 5 Java Class Library

Setelah itu klik tombol Next. Setelah itu akan keluar dialog seperti di bawah ini :



Gambar 6 New Java Class Library

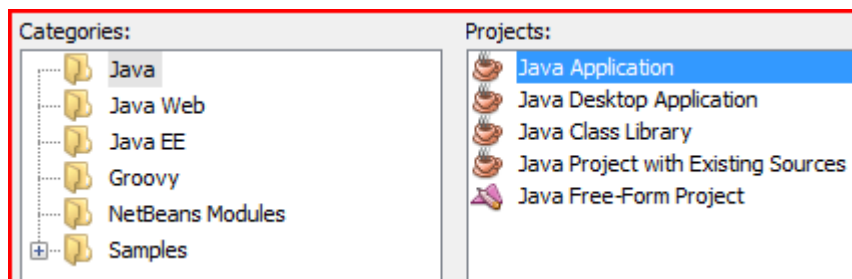
Pada bagian Project Name beri nama projectnya dengan nama “say-hello-api” atau terserah yang Anda sukai, setelah itu ubah Project Location sesuai dengan yang Anda mau. Jika selesai, klik tombol Finish :



Gambar 7 Project Manager

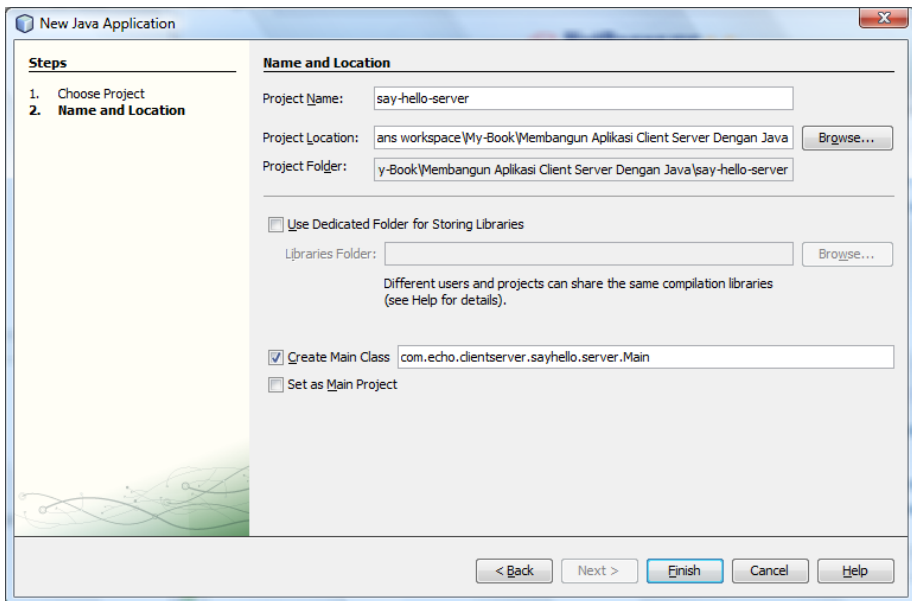
Sekarang anda bisa melihat project baru dengan nama “say-hello-api” pada tab Project NetBeans seperti terlihat pada gambar diatas.

Sekarang kita buat project kedua yaitu project “say-hello-server” yang kita gunakan sebagai project Server. Caranya sama, buat project baru lewat menu **File -> New Project** namun pada dialog pemilihan project pilih categories Java dan projects Java Application :



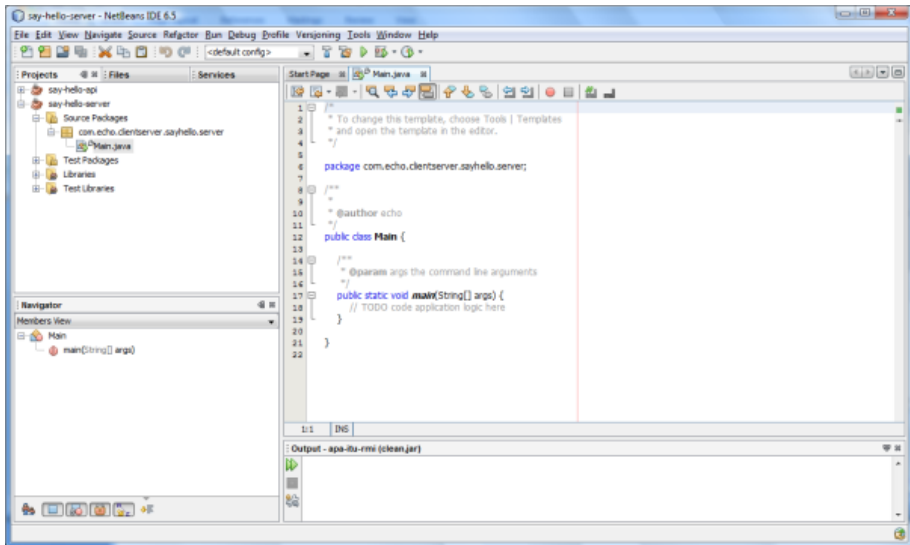
Gambar 8 Java Application

Setelah itu klik tombol Next, maka akan terlihat dialog seperti ini :



Gambar 9 New Java Application

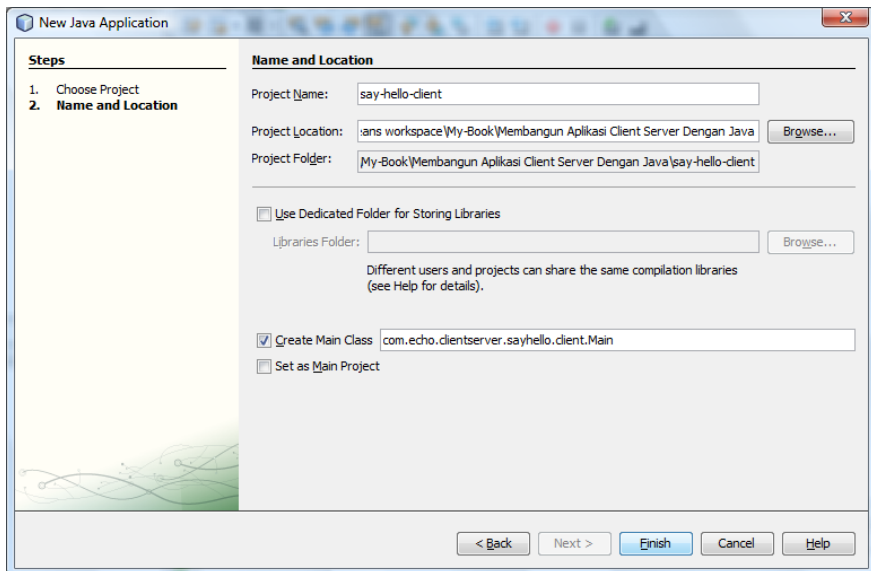
Pada bagian project name beri nama projectnya dengan nama “say-hello-server” dan pada bagian Create Main Class, ceklis checkbox itu lalu beri class utamanya dengan “com.echo.clientserver.sayhello.server.Main”. Maksud Creat Main Class ini adalah NetBeans akan otomatis membuat class Utama yang akan digunakan untuk menjalankan Aplikasi atau project yang akan kita buat. Setelah itu klik Finish :



Gambar 10 Tampilan Project SayHello Server

Setelah mengklik tombol Finish, maka otomatis NetBEans akan membuat project say-hello-server sekaligus class utama dan otomatis membuka class utama tersebut seperti terlihat pada gambar diatas.

Untuk terakhir pada proses pembuatan project Client, cara membuatnya sama seperti project server, hanya yang berbeda pada bagian penamaan project dan pembuatan class utama project seperti terlihat dibawah ini :

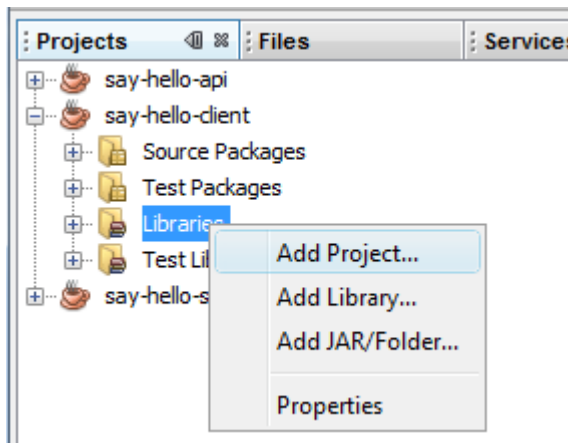


Gambar 11 New Java Application

Pada bagian project name, beri nama project dengan nama “say-hello-client” dan beri nama class utama dengan nama “com.echo.clientserver.sayhello.client.Main”.

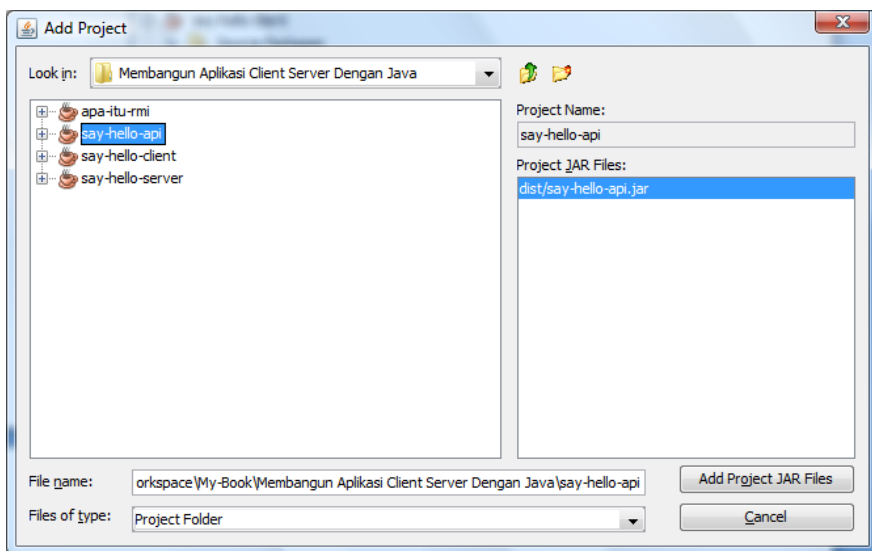
Setelah project Client selesai, sekarang kita telah mempunyai tiga project, say-hello-api, say-hello-server dan say-hello-client. Dan pada tahap terakhir kita harus menghubungkan antara project say-hello-api dengan project say-hello-server dan say-hello-client, hal ini harus dilakukan agar project say-hello-client dan say-hello-server dapat mengakses seluruh class dan interface dalam project say-hello-api.

Untuk melakukannya kita masuk kebagian Libraries project say-hello-server, lalu klik kanan dan pilih Add Project :



Gambar 12 Menambah Project

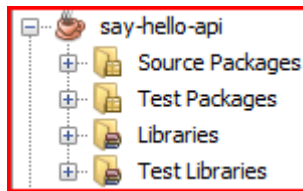
Lalu ketika keluar dialog seperti dibawah ini, pilih project say-hello-api :



Gambar 13 Pilih Project say-hello-api

Lalu klik Add Project JAR Files. Dan sekarang kita bisa mengakses project say-hello-api dari say-hello-server. **Lakukan hal yang sama pada project say-hello-client.**

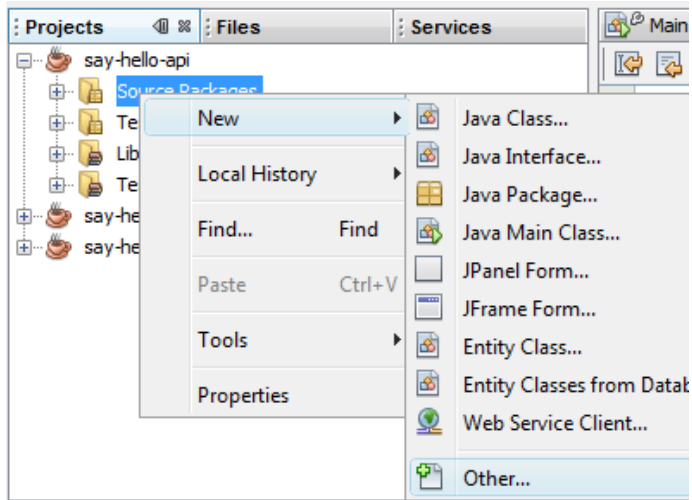
Sekarang saya akan bahas sedikit tentang bagian-bagian dari Project NetBEans yang terlihat dalam gambar dibawah ini :



Gambar 14 Bagian-Bagian Project

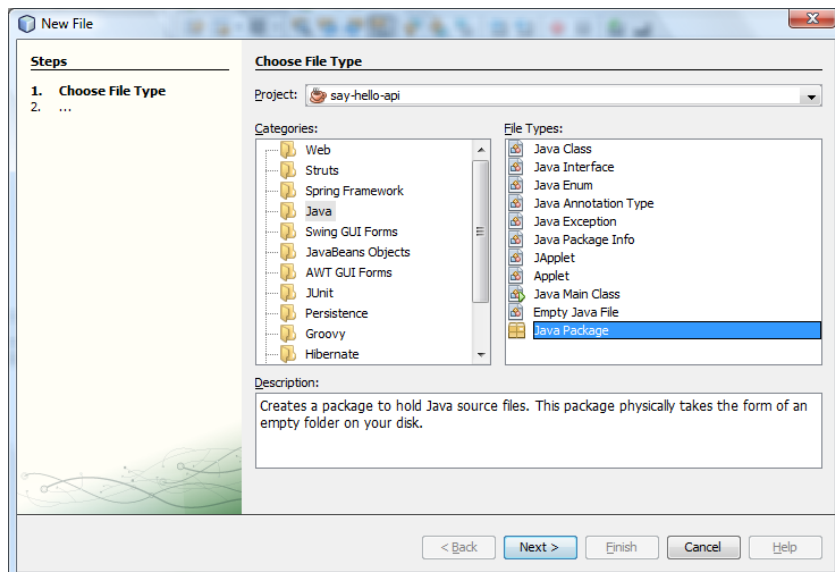
- **Source Package** merupakan tempat dimana kita membuat class dan interface, dalam bagian ini kita akan membuat aplikasi yang akan kita buat. Pada bagian ini kita tidak hanya harus membuat class dan interface, tapi kita juga bisa menyimpan file lain.
- **Test Package** merupakan tempat dimana kita melakukan proses testing. Dalam buku ini kita tidak akan menyentuh bagian ini karena Test Package hanya berguna sebagai proses testing menggunakan JUnit, untuk lebih jelas tentang JUnit bisa kunjungi <http://www.junit.org/>
- **Libraries** merupakan tempat library-library yang dibutuhkan oleh project yang kita buat (bagian Source Package), contohnya kita tadi menambahkan project say-hello-api ke library say-hello-server dan say-hello-client yang artinya kita akan menjadikan project say-hello-api sebagai library project say-hello-server dan say-hello-client.
- **Test Libraries** merupakan tempat library-library yang dibutuhkan oleh bagian Test Packages.

Sekarang kita buat package baru untuk project say-hello-api, caranya tinggal klik kanan pada bagian Source Package project say-hello-api lalu pilih **New -> Other**



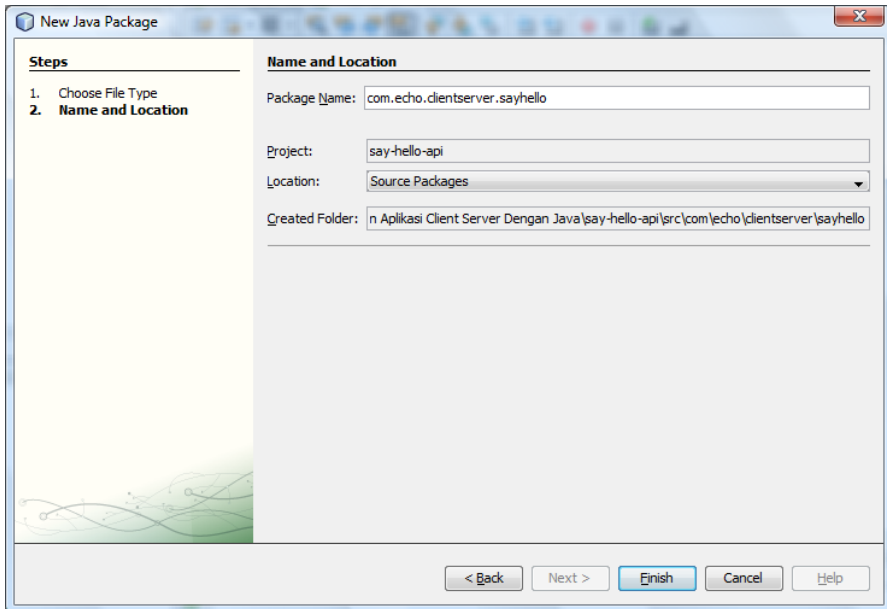
Gambar 15 New Package

Lalu akan keluar dialog seperti dibawah ini, setelah keluar dialog seperti dibawah pilih Categories Java dan File Types Java Package :



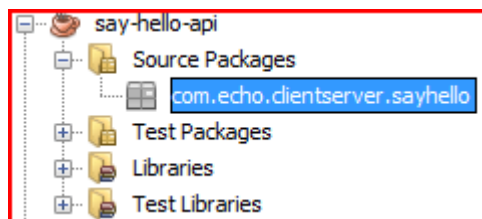
Gambar 16 New File

Lalu klik next dan akan terlihat dialog seperti dibawah ini :



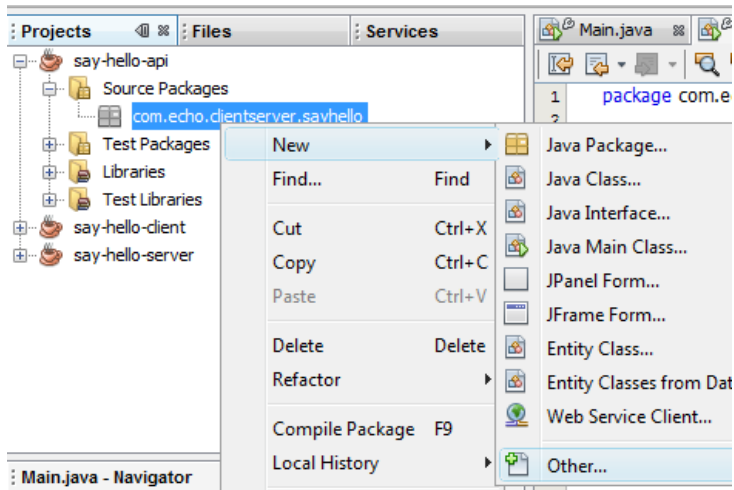
Gambar 17 New Java Package

Ubah nama package pada bagian Package Name menjadi “com.echo.clientserver.sayhello”, lalu klik Finish, sekarang kita bisa liat ada package baru di Source Package project say-hello-api :



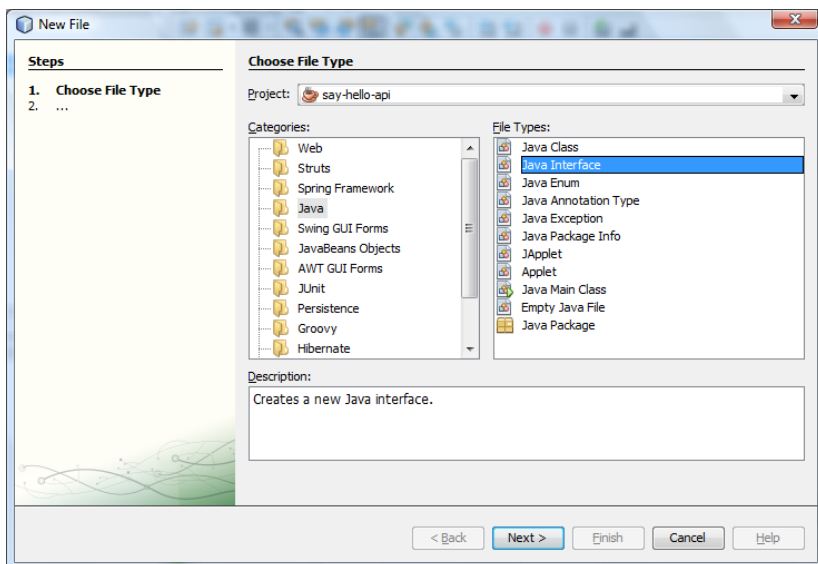
Gambar 18 Package Baru

Setelah membuat package, sekarang kita akan buat Interface SayHello seperti pada bagian awal bab ini, caranya klik kanan package yang tadi kita buat lalu pilih **New -> Other** :



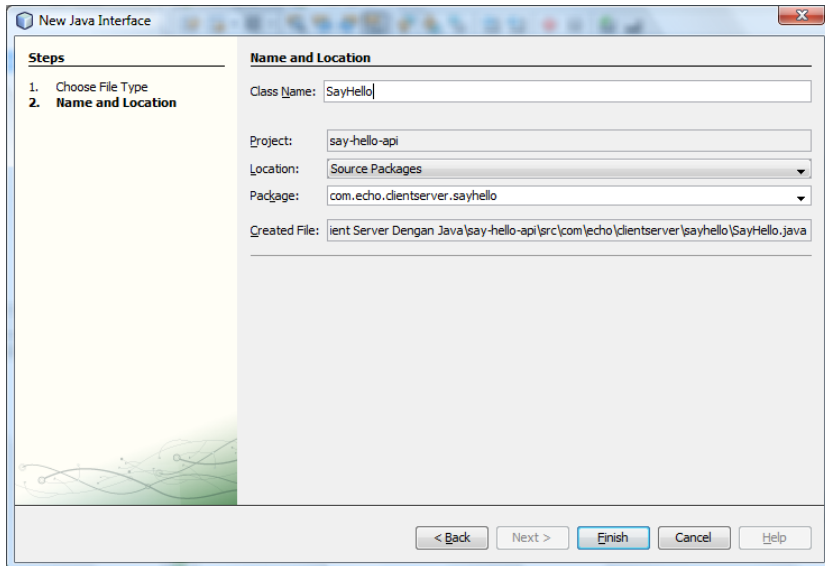
Gambar 19 New Interface

Setelah itu akan keluar dialog seperti dibawah ini, lalu pada categories pilih Java dan pada file types pilih Java Interface :



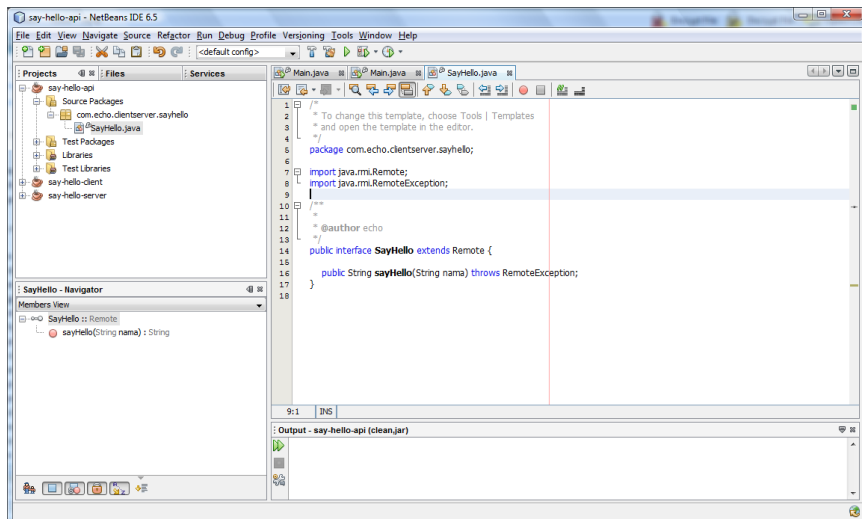
Gambar 20 New File

Lalu klik Next, dan akan terlihat dialog seperti dibawah ini :



Gambar 21 New Java Interface

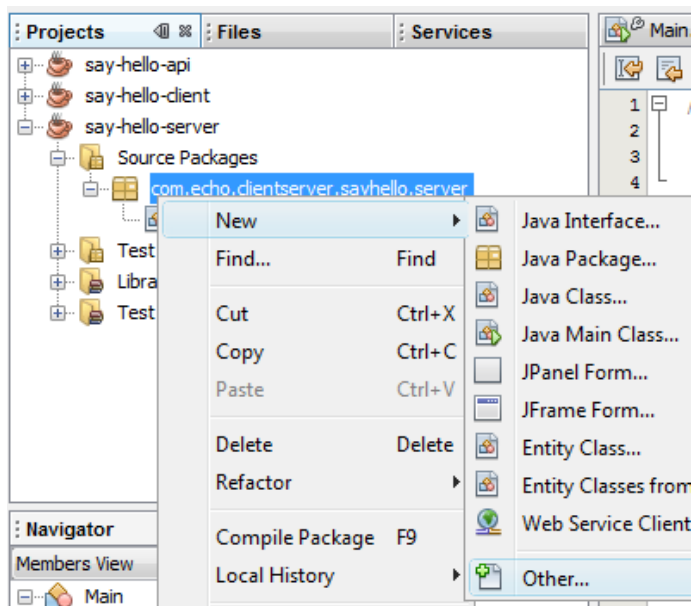
Beri nama Interface pada bagian Class Name dengan nama SayHello dan pastikan pada bagian package harus com.echo.clientserver.sayhello. Setelah selesai klik Finish :



Gambar 22 Interface SayHello

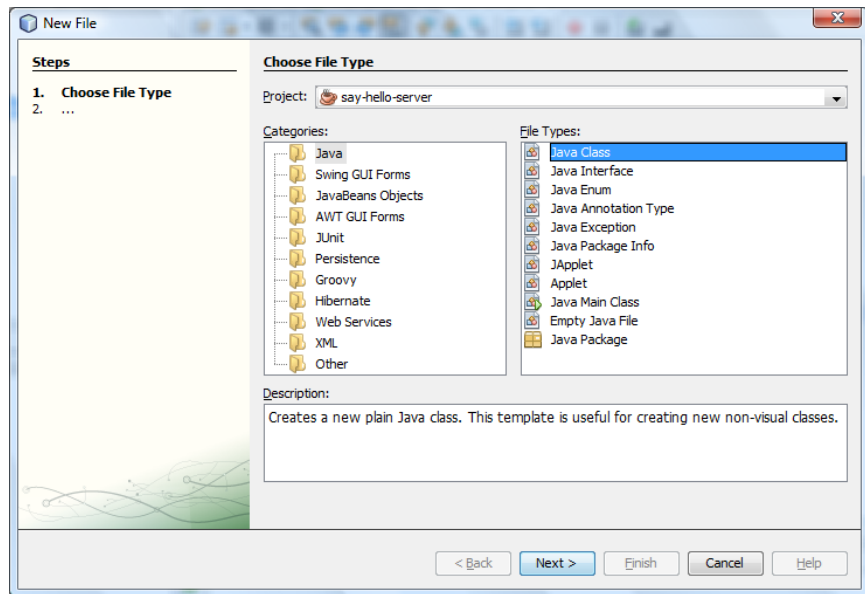
Pada interface SayHello ini, isikan seperti kode interface SayHello sebelumnya.

Sekarang kita akan membuat class SayHelloServer seperti pada kode sebelumnya. Seperti yang telah saya jelaskan tadi bahwa SayHelloServer ini akan ditempatkan di project say-hello-server. Untuk membuat class baru klik kanan pada package com.echo.clientserver.sayhello.server lalu pilih **New -> Other :**



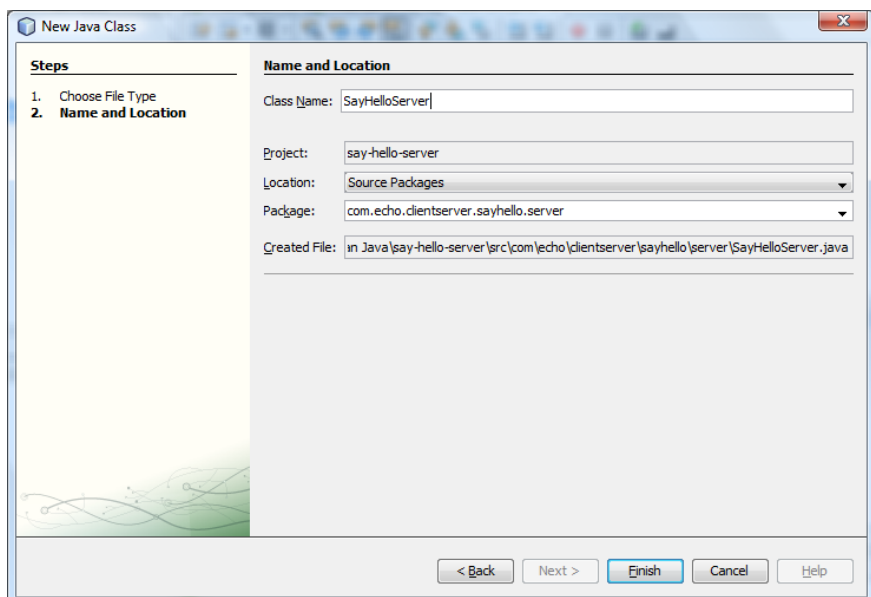
Gambar 23 New Class

Lalu akan keluar dialog seperti dibawah ini, lalu pilih categories Java dan file types Java Class :



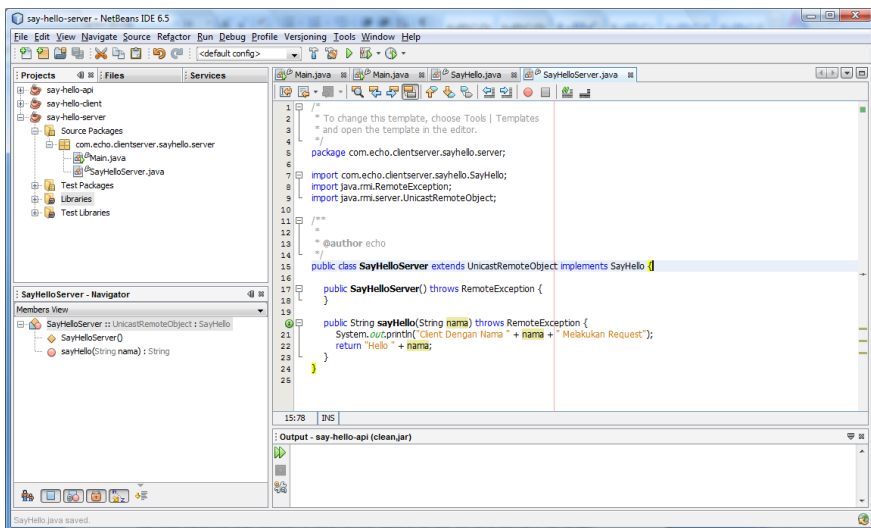
Gambar 24 New File

Setelah itu klik next dah akan terlihat dialog seperti dibawah ini :



Gambar 25 New Java Class

Jangan lupa beri nama class dengan nama SayHelloServer seperti pada kode tadi sebelumnya, setelah itu klik Next :



Gambar 26 Class SayHelloServer

Setelah itu isi class SayHelloServer dengan kode seperti sebelumnya telah kita buat pada awal bab ini.

Membuat Server SayHello

Untuk membuat Server SayHello, kita akan melakukannya di class Main yang tadi otomatis dibuat oleh NetBeans. Dan tentunya Server ini kita buat pada project say-hello-server. Seperti biasa, untuk membuat server kita berarti membuat sebuah Registry, setelah itu simpan object yang akan di akses lewat RMI di Registry :

```
package com.echo.clientserver.sayhello.server;
```

```
import java.rmi.RemoteException;  
import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;
```

```
public class Main {
```

```
public static void main(String[] args) throws RemoteException {  
    Registry registry = LocateRegistry.createRegistry(1099);  
  
    SayHelloServer sayHello = new SayHelloServer();  
    registry.rebind("sayHello", sayHello);  
  
    System.out.println("Server Telah Berjalan");  
}  
}
```

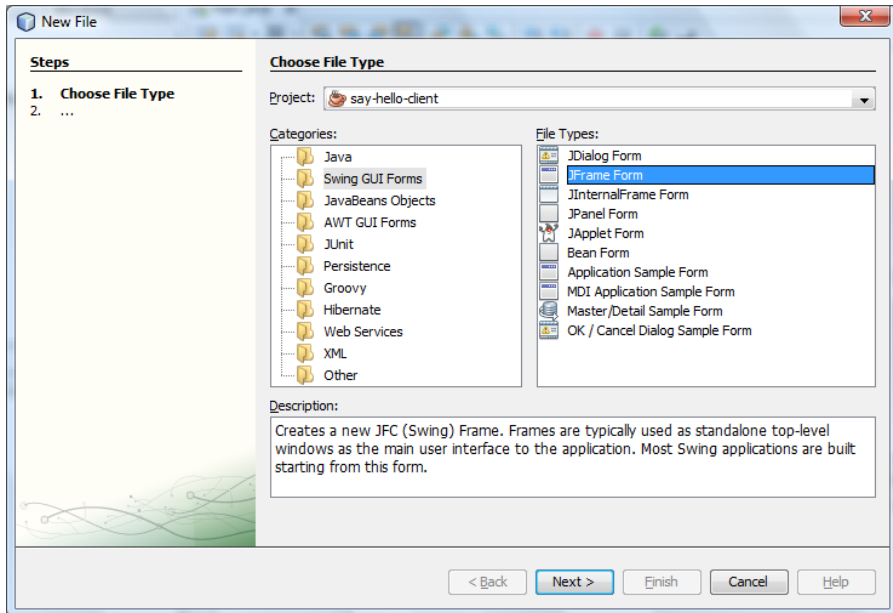
Kode 20 Class Main Server

Untuk membuat Server memang tidak terlalu rumit, karena tidak perlu membuat tampilan GUI nya, berbeda dengan Client, biasanya di sisi Client kita harus membuat tampilan GUI agar mempermudah User dalam menjalankan Aplikasinya.

Membuat Client SayHello

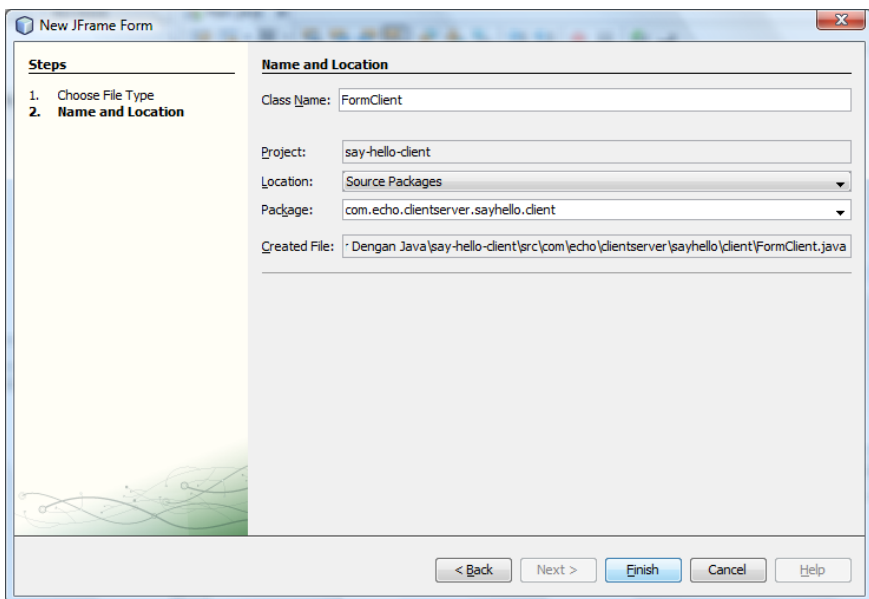
Karena pada sisi Client kita ingin dalam bentuk tampilan GUI, maka pertama kali kita harus membuat Form atau kalau dalam Java bernama JFrame. Dalam buku ini saya tidak akan membahas tentang Java GUI, oleh karena itu pada bagian Peringatan, saya menyarankan anda untuk menguasai Java GUI.

Untuk membuat JFrame dengan NetBeans cukup klik kanan package com.echo.clientserver.sayhello.client pada project say-hello-client lalu pilih **New -> Other**, setelah keluar tampilan seperti dibawah ini, pilih Categories Swing GUI Form dan File Types JFrame Form :



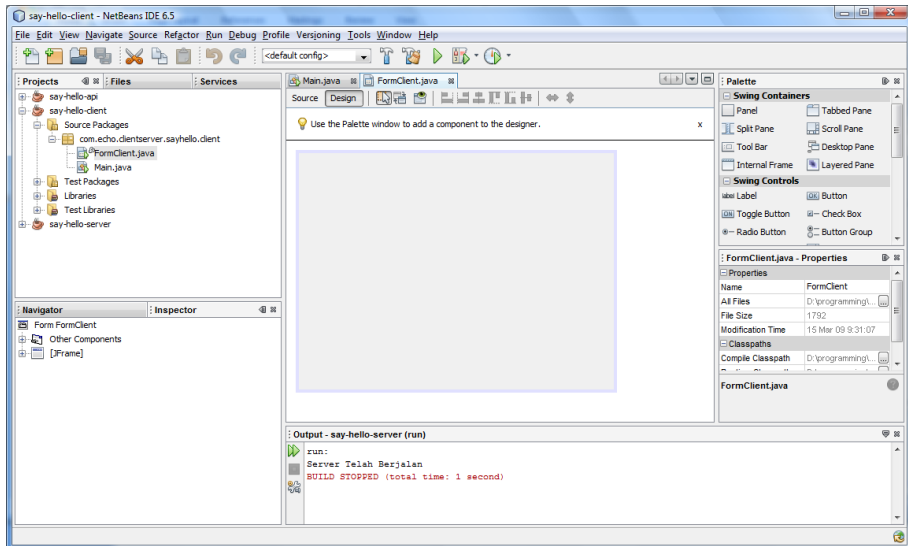
Gambar 27 New File

Setelah itu klik Next dan setelah keluar dialog seperti dibawah ini :



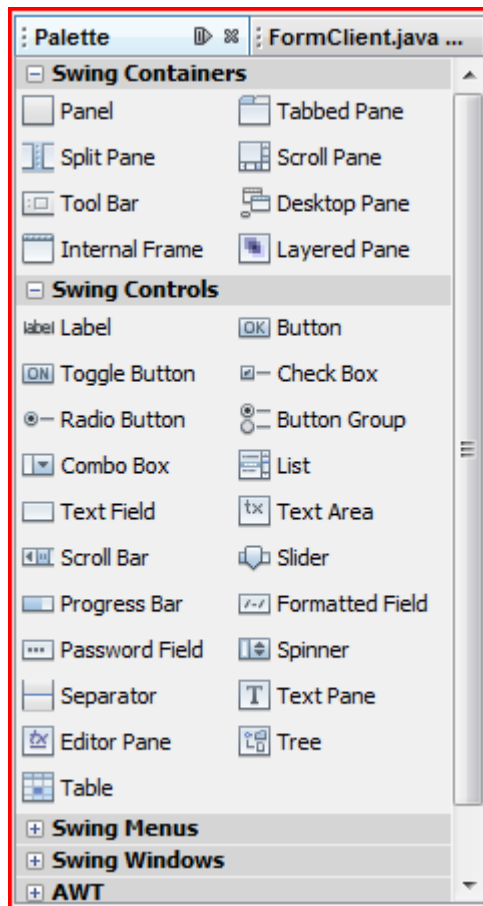
Gambar 28 New JFrame Form

Beri nama pada bagian Class Name dengan nama “FormClient” seperti terlihat pada gambar diatas. Setelah itu klik Finish. Dan sekarang NetBeans akan menampilkan GUI Builder yang bisa kita gunakan untuk membuat aplikasi berbasis GUI :

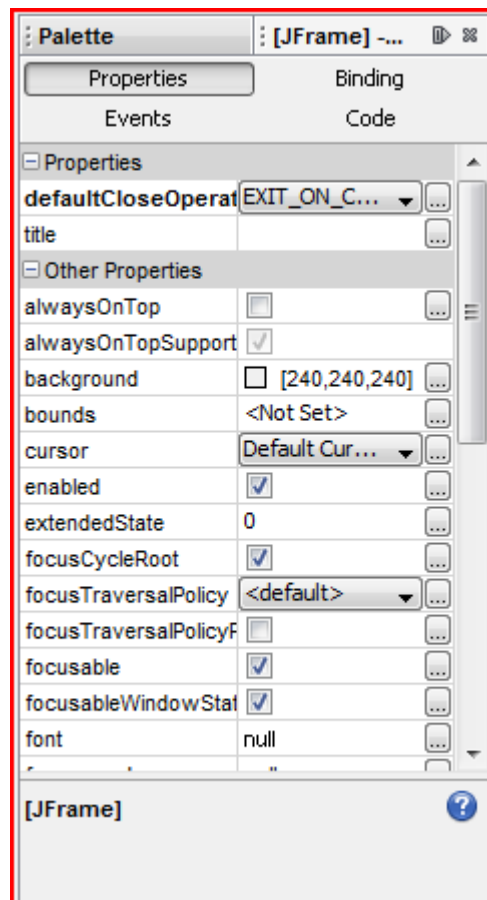


Gambar 29 GUI Builder pada NetBeans IDE

Pada bagian Pallet, Anda bisa mendrag dan drop komponen yang anda butuhkan ke JFrame, dan pada bagian properties, anda bisa mengubah properti-properti komponen yang sedang dalam fokus :

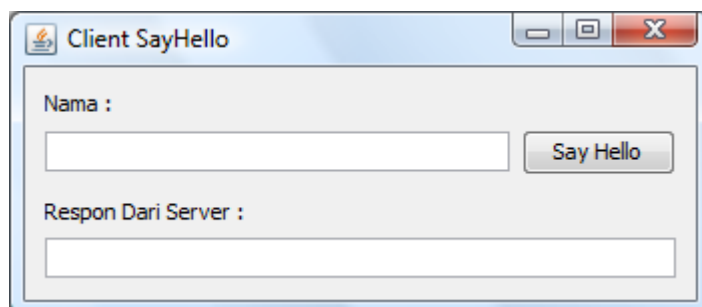


Gambar 30 Palette



Gambar 31 Properties

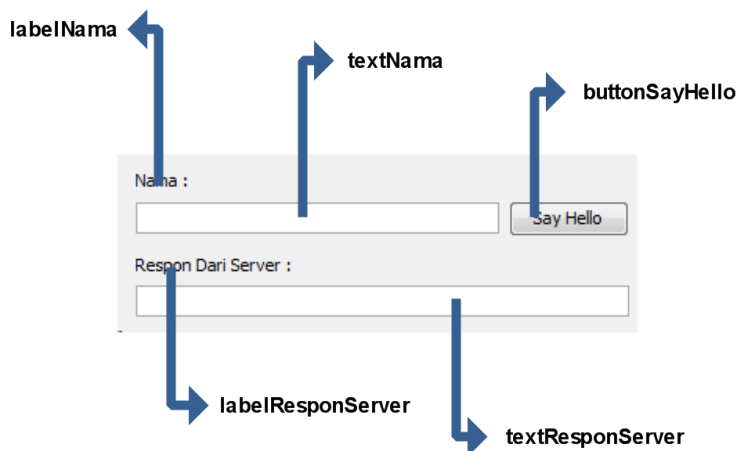
Buatlah tampilan JFrame tadi menjadi seperti ini :



Gambar 32 Tampilan Aplikasi SayHello

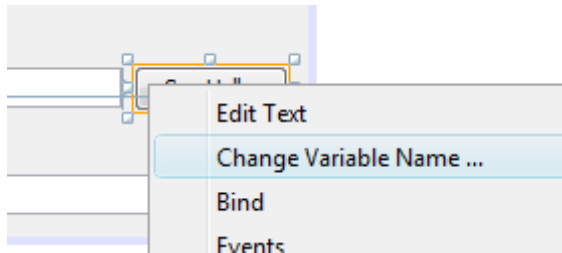
Saya tidak akan membahas bagaimana membuat tampilan seperti diatas, karena memang jika Anda sedang membaca buku ini berarti anda telah mengerti tentang Java GUI, selain itu jika anda belum terbiasa dengan NetBeans dalam membuat aplikasi GUI, anda hanya cukup mendrag komponen yang ada di Pallete ke dalam JFrame yang ada pada editor. Cukup mudah bukan?

Setelah itu ubah nama-nama variabel komponen tersebut menjadi seperti dibawah ini :



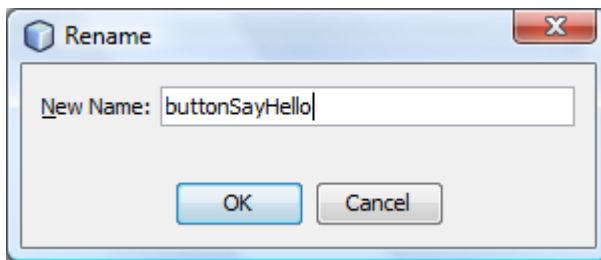
Gambar 33 Nama-Nama Variabel Form Client SayHello

Untuk merubah nama variabel komponen tersebut caranya dengan mengklik kanan pada komponen yang akan diubah nama variabelnya setelah itu klik Change Variable Name :



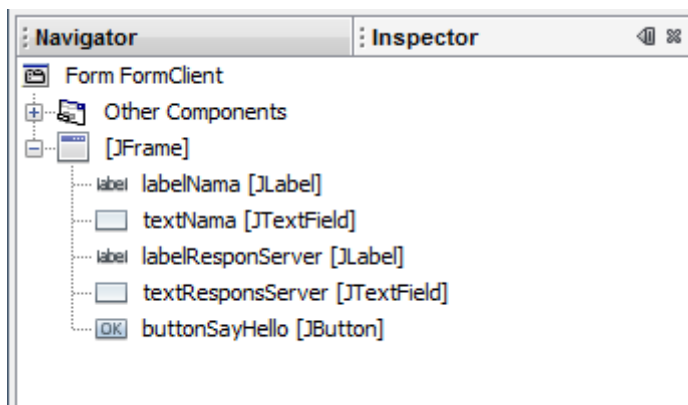
Gambar 34 Mengubah Nama Variabel

Setelah itu masukkan nama variabel baru pada dialog Rename :



Gambar 35 Rename

Setelah itu klik OK untuk menyimpan perubahan. Jika Anda telah mengubah seluruh variabel komponen diatas, maka pada Inspector akan terlihat seperti ini :



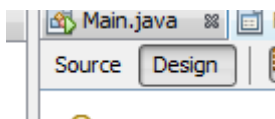
Gambar 36 Inspector

Sekarang tahap terakhir adalah membuat koneksi dari Client ke Server dengan cara hampir mirip dengan bab sebelumnya :

```
SayHello sayHello =  
(SayHello) Naming.lookup("rmi://localhost:1099/sayHello");
```

Kode 21 Membuat Registry Client

Ketikkan kode diatas pada konstruktor FormClient yang tadi kita buat, caranya masuk ke bagian Source :



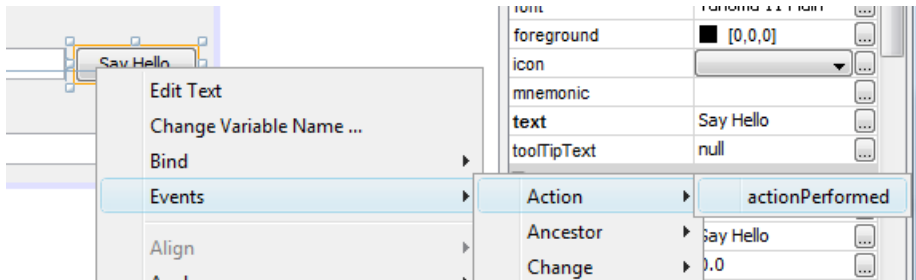
Gambar 37 Button Source

Setelah itu ubah konstruktornya sehingga menjadi seperti pada gambar dibawah :

```
private SayHello sayHello;  
  
/** Creates new form FormClient */  
public FormClient() {  
    try {  
        sayHello = (SayHello) Naming.lookup("rmi://localhost:1099/sayHello");  
    } catch (NotBoundException ex) {  
        System.out.println("Terjadi Error Dengan Pesan : " + ex.getMessage());  
        System.exit(1);  
    } catch (MalformedURLException ex) {  
        System.out.println("Terjadi Error Dengan Pesan : " + ex.getMessage());  
        System.exit(1);  
    } catch (RemoteException ex) {  
        System.out.println("Terjadi Error Dengan Pesan : " + ex.getMessage());  
        System.exit(1);  
    }  
  
    initComponents();  
}
```

Gambar 38 Konstruktor Form Client

Setelah itu kembali lagi ke bagian Design dengan mengklik tombol Design, lalu beri ActionListener pada buttonSayHello dengan cara klik kanan buttonSayHello lalu pilih **Events -> Action -> actionPerformed** :



Gambar 39 Menambah ActionListener

Setelah itu maka otomatis akan terbuat sebuah metode baru yang digunakan sebagai metode yang akan dijalankan ketika tombol diklik :

```
private void buttonSayHelloActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

Gambar 40 metode buttonSayHelloActionPerformed

Sekarang kita tinggal memberi aksi pada metode buttonSayHelloActionPerformed diatas :

```
String nama = textNama.getText();  
String response = sayHello.sayHello(nama);  
textResponServer.setText(response);
```

Kode 22 Isi metode buttonSayHelloActionPerformed

Dalam aksi diatas, pertama kita menampung isi teks yang ada dalam textNama kedalam variabel nama, setelah itu kita melakukan pemanggilan metode sayHello(nama) dan datanya ditampung pada variabel response, setelah itu data response tersebut ditampilkan di textResponServer.

Sehingga metode buttonSayHelloActionPerformed menjadi seperti pada gambar dibawah ini :

```
private void buttonSayHelloActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String nama = textNama.getText();
        String response = sayHello.sayHello(nama);
        textResponsServer.setText(response);
    } catch (RemoteException ex) {
        System.err.println("Terjadi Error Dengan Pesan : " + ex.getMessage());
        System.exit(1);
    }
}
```

Gambar 41 Isi metode buttonSayHelloActionPerformed

Sekarang proses pembuatan Client Selesai, tinggal kita ubah clas Utama (Main) yang tadi dibuatkan oleh NetBeans untuk project say-hello-client menjadi seperti ini :

```
package com.echo.clientserver.sayhello.client;

import javax.swing.SwingUtilities;

public class Main {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {

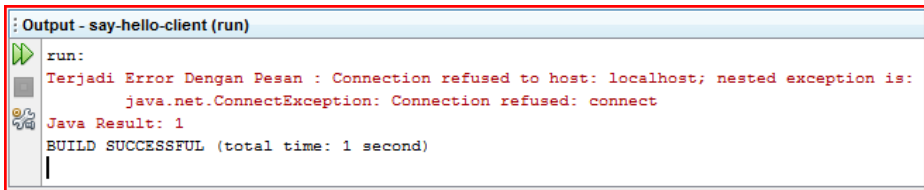
            public void run() {
                FormClient client = new FormClient();
                client.setVisible(true);
            }
        });
    }
}
```

Kode 23 Class Main Client

Menjalankan Aplikasi SayHello

Setelah tadi kita membuat Client dan Server untuk Aplikasi SayHello, sekarang saatnya kita jalankan Aplikasi SayHello tersebut, tapi untuk pertama kali, jalankan terlebih dahulu Server-nya, karena jika Client

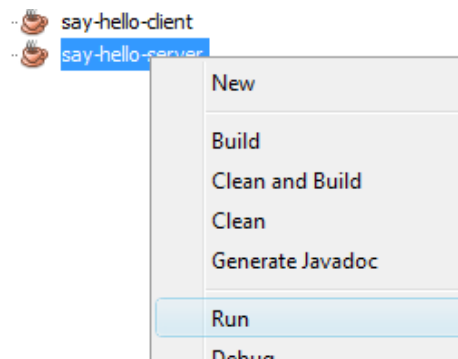
dijalankan terlebih dahulu maka akan terjadi Error seperti terlihat dibawah ini :



```
Output - say-hello-client (run)
run:
Terjadi Error Dengan Pesan : Connection refused to host: localhost; nested exception is:
    java.net.ConnectException: Connection refused: connect
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

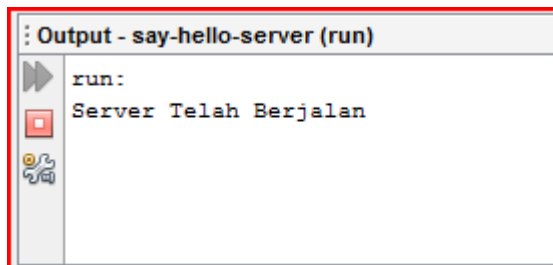
Gambar 42 Error Client ketika dijalankan

Jadi untuk pertama kali kita harus jalankan Server, caranya klik kanan project say-hello-server lalu klik Run :



Gambar 43 Menjalankan Server

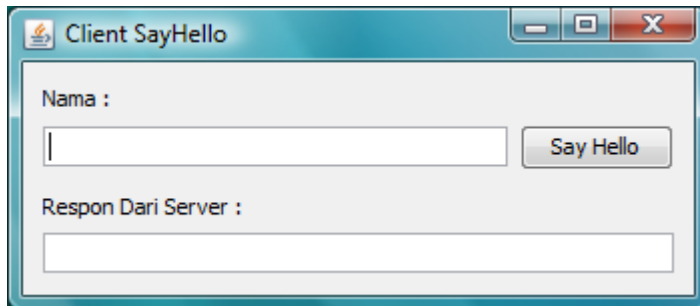
Jika Server berhasil berjalan, maka akan terlihat output trace seperti pada gambar dibawah ini :



```
Output - say-hello-server (run)
run:
Server Telah Berjalan
```

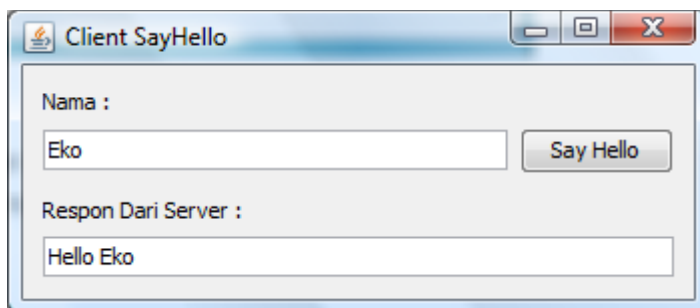
Gambar 44 Output Trace Server

Artinya Server telah Berjalan, sekarang tinggal kita jalankan Client, caranya sama seperti menjalankan Server, klik kanan project say-hello-client, lalu pilih Run, maka akan keluar Aplikasi seperti dibawah ini :



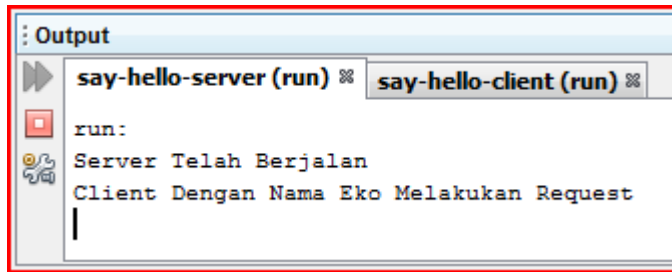
Gambar 45 Aplikasi Client

Sekarang coba masukkan tulisan “Eko” di textNama, lalu klik buttonSayHello, maka server akan merespon dan hasil responnya ditampilkan di textResponServer :



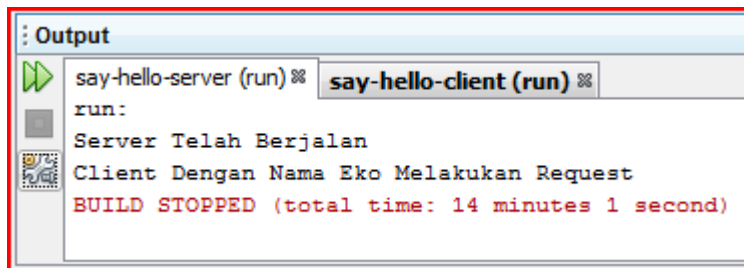
Gambar 46 Response dari Server

Dan sekarang kita bisa lihat trace responnya di output Server :



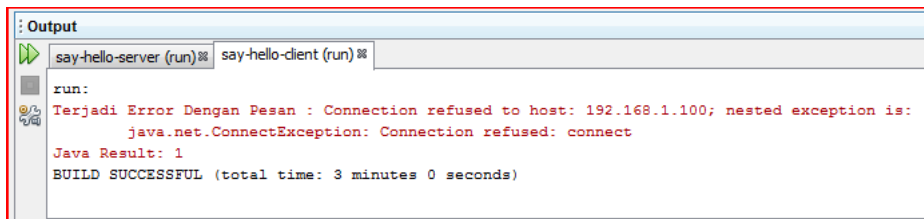
Gambar 47 Trace Respon Server

Ini membuktikan bahwa Client saling berkomunikasi dengan Server, tidak percaya? Coba Anda matikan server dengan mengklik tombol kotak merah seperti yang terlihat diatas sehingga menjadi seperti ini ini :



Gambar 48 Mematikan Server

Setelah itu klik kembali buttonSayHello pada aplikasi Client SayHello, maka program akan keluar dan pada trace Client akan terlihat seperti Ini :



Gambar 49 Error pada Client ketika Server mati

Ini berarti memang Client saling berkomunikasi dengan Server, sehingga jika Server mati, maka Client pun akan ikut mati.

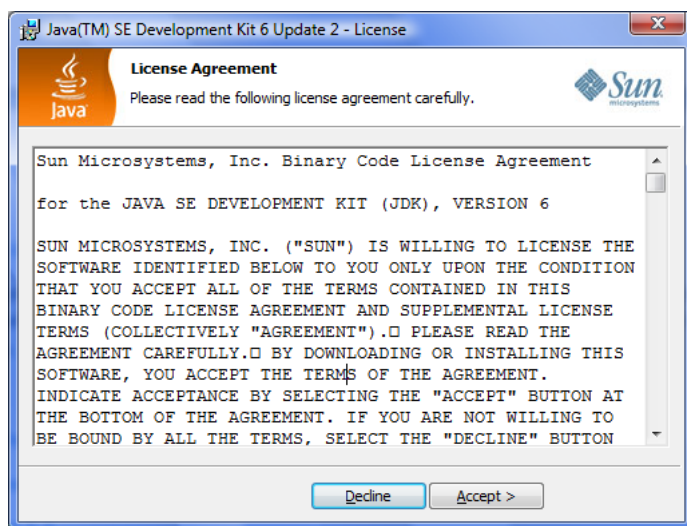
Dan sekarang anda telah membuat sebuah aplikasi Client Server yang saling merespon dari Client dan Server, berbeda pada bab sebelumnya yang hanya Client memerintah Server, pada aplikasi ini kita telah belajar tentang aplikasi yang dimana Client melakukan request dan Server merespon.

Sebenarnya inti dari membangun aplikasi client server hanyalah seperti itu, hanya saja kadang semakin rumit aplikasi yang dibuat, semakin rumit pula cara membuatnya. Sehingga pelajaran tadi belum cukup agar Anda dapat membuat aplikasi client server yang rumit.

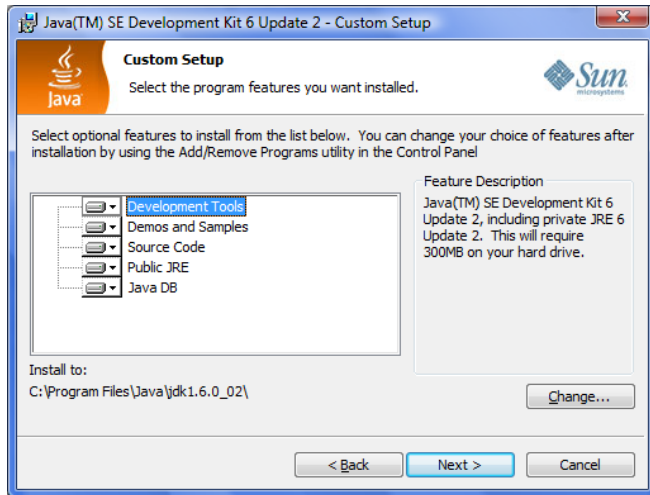
Lampiran

Instalasi JDK

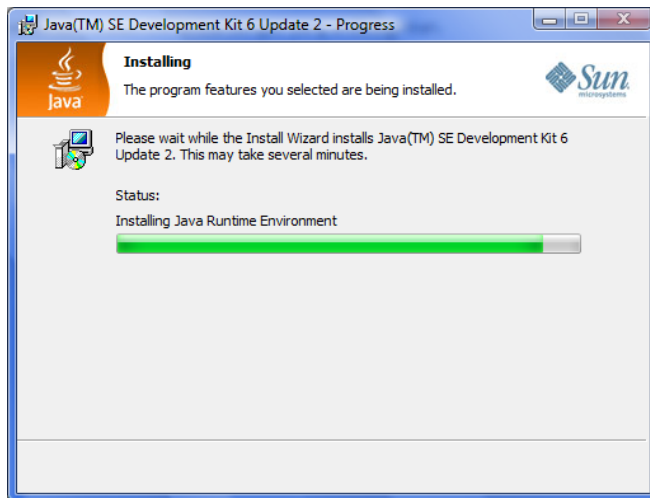
Sebelum Anda membuat program berbasis Java, pastilah harus terinstal JDK dalam komputer Anda.



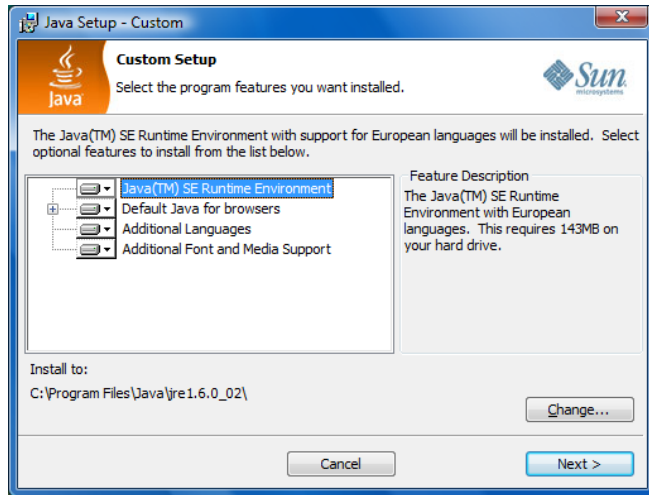
Gambar diatas adalah tampilan awal proses instalasi JDK, klik Accept untuk melanjutkan proses instalasi.



Tentukan lokasi instalasi JDK sesuai dengan yang anda inginkan.



Gambar diatas menampilkan proses instalasi yang sedang berjalan setelah selesai menginstal JDK maka otomatis akan terinstal pula JRE, sehingga muncul kotak dialog seperti dibawah.



Tentukan lokasi JRE sesuai dengan yang anda inginkan lalu klik Next.



Tunggu sampai JRE selesai terinstal. Setelah selesai maka akan terlihat dialog seperti dibawah ini.



Tentang Saya



Saya lebih senang mengenalkan diri saya sendiri, oleh karena itu saya beri judul pada bagian ini “Tentang Saya” bukan “Tentang Penulis”.

Nama saya Eko Kurniawan Khannedy, tapi kebanyakan teman-teman saya memanggil saya Usu, tapi kalau di dunia maya saya lebih dikenal dengan panggilan echo_khannedy. Saya lahir di Bekasi, tapi entah kenapa di akte kelahiran tertulis Subang 😊 tepatnya pada

tanggal 29 Desember 1988. Saat ini saya sedang menempuh kuliah di UNIKOM Bandung di jurusan Teknik informatika.

Walaupun di UNIKOM Bandung saya tidak mendapat mata kuliah pemrograman Java, tapi saya sangat menyukai pemrograman Java. Sejak tahun 2007 sampai sekarang saya aktif menulis di wordpress, Anda bisa mengunjungi wordpress saya di <http://eecchhoo.wordpress.com/>. Selain sibuk menulis di wordpress, saya juga aktif di milis-milis seperti milis Jug-Bandung, Jug-Joglosemar, NetBeans Indonesia, dan lain-lain.

Anda bisa menghubungi saya lewat email di echo.khannedy@gmail.com atau lewat yahoo messenger di *echo.khannedy* atau lewat hp di 085292775999.