Remote Method Invocation di Java

Remote Method Invocation (RMI) adalah suatu API yang memungkinkan suatu obyek memanggil atau meminta eksekusi suatu metode pada obyek yang ada di dalam ruang alamat yang lain (beda proses atau progam), yang dapat berada pada mesin yang sama atau pada mesin berbeda (dan jauh). Melalui RMI, obyek yang berjalan dalam suatu JVM pada suatu komputer (sisi Client) dapat meminta eksekusi metode pada obyek yang hadir di JVM lain (sisi Server). RMI membuat suatu obyek Server jauh bersifat publik sehingga memungkinkan komunikasi di sisi client dan server dengan mudah melalui pemanggilan metode pada obyek server.

Cara Kerja RMI

Komunikasi antara client dan server ditangani dengan menggunakan dua obyek penengah (*intermediate*), yaitu obyek Stub (pada sisi client) dan obyek Skeleton (pada sisi server).

Obyek Stub

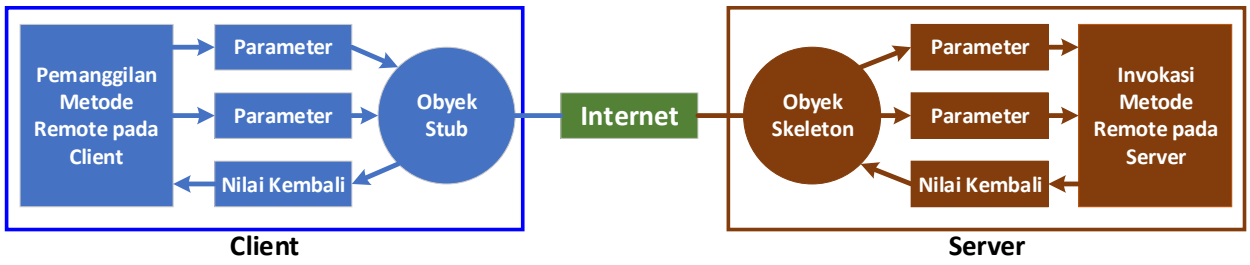
Obyek stub pada mesin client membangun suatu blok informasi dan mengirimkan informasi ini ke server. Blok ini terdiri dari:

- Suatu identifier dari obyek jauh (*remote*)yang akan digunakan
- Name metode yang akan dieksekusi
- Parameter-parameter untuk JVM *remote* tersebut.

Obyek Skeleton

Obyek skeleton mengirimkan request yang diterima dari obyek stub ke obyek remote (RMI) yang dituju. Skeleton mengerjakan tugas berikut:

- Memanggil metode yang diinginkan pada obyek ril yang hadir di server.
- Meneruskan parameter-parameter yang diterima dari obyek Stub ke metode tersebut.



Langkah-langkah Mengimplementasikan Interface

1. Mendefinisikan suatu antarmuka jauh (*remote interface*)
2. Mengimplementasikan *remote interface* tersebut
3. Membuat obyek Stub dan Skeleton dari kelas implementasi menggunakan rmic (rmi complier)
4. Memulai rmiregistry
5. Membuat dan mengeksekusi program aplikasi server
6. Membuat dan mengeksekusi program aplikasi client.

Langkah 1: Mendefinisikan *remote interface*

Hal pertama yang harus dilakukan adalah membuat suatu interface yang akan menyediakan deskripsi dari metode yang dapat diinvokasi oleh remote client. Interface ini harus meng-extend Remote interface dan prototipe metode di dalam interface tersebut harus men-throw RemoteException-nya.

```
// Membuat interface bernama Search
import java.rmi.*;

public interface Search extends Remote {
    // Mendeklarasikan prototipe dari metode
    public String query(String search) throws RemoteException;
}
```

Langkah 2: Mengimplementasikan *remote interface*

Langkah berikutnya adalah mengimplementasikan remote interface tersebut. Untuk melakukan ini, kelas tersebut harus meng-extend kelas UnicastRemoteObject dari paket java.rmi. Juga, suatu konstruktor default perlu dibuat untuk men-throw java.rmi.RemoteException dari konstruktor induknya di dalam kelas tersebut.

```
// Program Java untuk mengimplementasikan interface bernama Search
import java.rmi.*;
import java.rmi.server.*;

public class SearchQuery extends UnicastRemoteObject implements Search {
    // Konstruktor default untuk men-throw RemoteException
    // dari konstruktor induknya
    SearchQuery() throws RemoteException {
        super();
    }

    // Implementasi dari interface query
    public String query(String search) throws RemoteException {
        String result;
        if (search.equals("Pemrograman Terdistribusi dengan Java"))
            result = "ditemukan";
        else
```

```
            result = "tidak ditemukan";

        return result;
    }
}
```

Langkah 3: Membuat Obyek Stub dan Skeleton dari Kelas Implementasi menggunakan rmic

Tool rmic digunakan untuk meng-invokasi compiler rmi yang membuat obyek Stub dan Skeleton. Prototipenya adalah nama kelas rmic. Untuk program di atas, perintah berikut perlu dieksekusi pada command prompt:

```
rmic SearchQuery
```

Langkah 4: Memulai rmiregistry

Layanan registry harus dimulai dengan menjalankan perintah berikut pada command prompt:

```
start rmiregistry
```

Langkah 5: Membuat dan Mengeksekusi Program Aplikasi Server

Langkah berikutnya adalah membuat program aplikasi server dan mengeksekusinya pada suatu command prompt terpisah.

- Program server ini menggunakan metode createRegistry dari kelas LocateRegistry untnuk membuat rmiregistry di dalam server JVM dengan nomor port yang dilewatkan sebagai argumen.
- Metode rebind dari kelas Naming digunakan untuk mengikat obyek remote ke nama yang baru.

```
//program untuk aplikasi server
import java.rmi.*;
import java.rmi.registry.*;

public class SearchServer {
    public static void main(String args[]) {
        try {
            // Membuat suatu obyek dari kelas implementasi interface
            Search obj = new SearchQuery();

            // rmiregistry di dalam server JVM dengan nomor port 2019
            LocateRegistry.createRegistry(2019);

            // Ikat obyek remote dengan nama sister2019
            Naming.rebind("rmi://localhost:2019"+ "/sister2019", obj);
        }
        catch(Exception ae) {
            System.out.println(ae);
```

```
        }
    }
}
```

Langkah 6: Membuat dan Mengeksekusi Program Aplikasi Client

Langkah terakhir adalah membuatkan program aplikasi client dan mengeksekusinya pada suatu command prompt terpisah. Metode lookup dari kelas Naming digunakan untuk mendapatkan referensi dari obyek Stub.

```
//program untuk aplikasi client
import java.rmi.*;

public class ClientRequest {
    public static void main(String args[]) {
        String answer, value = "Pemrograman Terdistribusi dengan Java";
        try  {
            // metode lookup untuk menemukan referensi dari obyek remote
            Search access =
                (Search)Naming.lookup("rmi://localhost:2019"+ "/sister2019");

            answer = access.query(value);
            System.out.println("Artikel pada mengenai " + value +
                            " " + answer + " pada server Sister2019.");
        }
        catch(Exception ae) {
            System.out.println(ae);
        }
    }
}
```

***Catatan**: program server dan client di atas dieksekusi pada mesin yang sama sehingga digunakan nama alamat localhost. Agar dapat mengakses obyek remote yang berada di mesin lain, localhost harus diganti dengan nama mesin atau IP address dimana obyek remote tersebut berjalan.*

Pengamatan Penting:

1. RMI merupakan solusi murni Java terhadap *Remote Procedure Calls* (RPC) dan digunakan untuk membuat aplikasi terdistribusi dengan Java.
2. Obyek Stub dan Skeleton digunakan untuk komunikasi antara sisi client dan server.