MEMBANGUN APLIKASI TERDISTRIBUSI DENGAN .NET REMOTING

Erick Kurniawan

Abstrak

Teknologi .NET Remoting merupakan salah satu teknologi sistem terdistribusi dari Microsoft yang memungkinkan penggunaan sebuah logic untuk aplikasi dengan basis yang berbeda. Pada .NET Remoting, logic untuk aplikasi desktop dapat juga diakses oleh aplikasi Web, sehingga hanya dibutuhkan sebuah logic yang sama. Jika logic hendak diubah, hanya perlu melakukan satu kali perubahan saja. .NET Remoting dapat diimplementasikan dengan menggunakan bahasa pemrograman C#, misalnya pada kasus registrasi mahasiswa UKDW dengan menggunakan sebuah objek remote yang sama untuk aplikasi client yang berbasis desktop dan Web.

Kata Kunci: *.NET Remoting, Sistem Terdistribusi*

1. Pendahuluan

Pada sistem *client server*, terdapat komponen-komponen yang memiliki peran yang berbeda, tetapi dapat saling bekerja sama. Komponen-komponen tersebut adalah *user interaction*, *application*, *database management*, dan *middleware*. Agar tercapai efektivitas, komponen-komponen tersebut didistribusikan, maka prosesor diberi tugas secukupnya, sehingga unjuk kerja seluruh sistem dapat dioptimalkan.

Web services merupakan salah satu cara yang paling sering digunakan untuk mengakses komponen eksternal. Namun jika komponen yang diperlukan merupakan komponen internal, penggunaan .NET Remoting akan menyediakan performa yang lebih baik. Dalam studi kasus registrasi pengambilan matakuliah mahasiswa, *client* hanya mengakses komponen internal.

.NET Remoting merupakan teknologi yang mengijinkan terjadinya komunikasi antara aplikasi *client* dan komponen. Sebagai hasil, *remotable components* akan menjadi lebih cepat daripada *Web services*. Sistem yang dibuat hanya akan memerlukan sebuah *logic* yang dapat digunakan baik oleh aplikasi *desktop* maupun aplikasi *Web*, sehingga tidak memerlukan perubahan dua *logic*. Cara pendistribusian yang digunakan adalah *thin client* yang berarti hanya *presentation* yang terdapat pada *client*. Aplikasi dan *database management* akan berada pada sisi *server*.

Tujuan dari penelitian ini adalah untuk mengimplementasikan teknologi .NET Remoting untuk membuat sebuah *logic* yang dapat digunakan oleh aplikasi *desktop* dan aplikasi *Web* ke dalam studi kasus registrasi pengambilan matakuliah mahasiswa UKDW.

2. Landasan Teori

a. Teknologi .NET Remoting

Remoting adalah proses dari program atau komponen yang berinteraksi melewati batas-batas tertentu (Ingo Rammer, 2002). Batas-batas tersebut adalah proses ataupun komputer yang berlainan. Implementasi dari *remoting* membedakan objek menjadi dua macam, yaitu objek *remote* dan objek *mobile*.

Objek *remote* memiliki kemampuan untuk mengeksekusi *method* pada *server* yang terpisah, mengirimkan parameter, dan menerima nilai (*return value*). Objek *remote* ini akan selalu berada pada sisi *server*, dan hanya *reference* yang akan dikirim ke komputer yang lain.

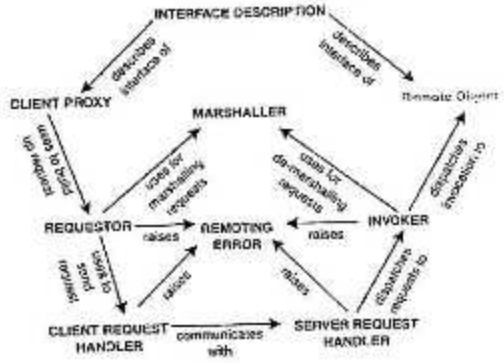
Pada objek *mobile*, saat objek tersebut dikirimkan ke proses atau komputer yang berlainan, objek tersebut akan di-*serialized* (*marshalled*) menjadi representasi yang umum dan akan di-*deserialized* pada tujuan. *Marshaller* bertugas untuk mengubah *remote*

invocation menjadi *byte stream*, sehingga dapat dikirimkan pada jaringan. *Server* dan *client* akan memiliki *copy* objek yang sama. *Method* yang dieksekusi pada objek *copy* tersebut akan selalu menjadi *context* lokal dan tidak ada *message* yang akan dikirim kembali ke tempat asal objek. Objek *copy* tidak akan dibedakan dari objek regular, dan tidak akan ada pembedaan antara objek *server* dan objek *client*.

Pola dasar dari *remoting* itu sendiri adalah seperti yang tampak pada Gambar 1. Untuk mengakses objek *remote*, *client* dapat langsung menggunakan *requestor* ataupun menggunakan sebuah *client proxy*. Sebuah *requestor* membangun *remote invocation* yang berisi parameter dari objek *remote*. Sedangkan *client proxy* merupakan objek lokal pada proses *client* yang memiliki *interface* yang sama dengan objek *remote*. *Interface* ini didefinisikan dengan menggunakan *interface description*. *Client proxy* kemudian menggunakan *requestor* untuk membangun *remote invocation*.

Requestor pada *client* menggunakan *client request handler* untuk komunikasi pada jaringan. Pada sisi *server*, *remote invocation* akan diterima oleh *server request handler* yang kemudian diteruskan ke *invoker*. Tugas *invoker* adalah membaca *request* kemudian memanggil *remote* objek yang diminta *client*.

Parameter yang diteruskan *client* dan *server* di-*serialized* dan *deserialized* menggunakan *marshaller*. Jika *client* mengalami masalah teknik dalam berkomunikasi atau *server* mengalami masalah internal, maka kesalahan tersebut akan diteruskan ke *client request handler* dengan menggunakan *remoting error*.



Gambar 1. Pola dasar remoting

Salah satu teknologi *remoting* dalam .NET adalah .NET Remoting yang merupakan pengganti DCOM. .NET Remoting menyediakan performa dan fleksibilitas yang bagus untuk sistem terdistribusi. Teknologi ini mengizinkan *client* untuk menggunakan komponen pada komputer yang terpisah dan menggunakannya sebagai komponen lokal. .NET Remoting merupakan pilihan yang ideal untuk aplikasi intranet (Matthew MacDonald, 2003) yang membutuhkan pendistribusian objek.

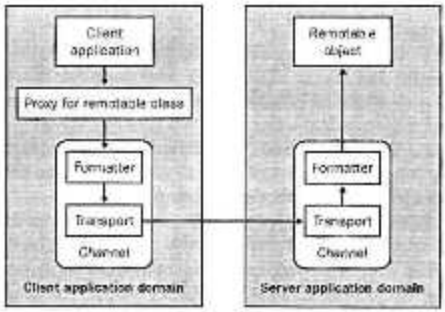
b. .NET Remoting

NET *Remoting* merupakan teknologi yang memungkinkan aplikasi pada *client* untuk menggunakan komponen pada komputer lain dan memakainya seperti komponen lokal (Ingo Rammer, 2002). .NET *Remoting* memungkinkan terjadinya komunikasi dengan sebuah objek

pada *app domain* yang berbeda. *App domain* merupakan area dimana semua aplikasi menggunakan *memory* yang sama (Richard Blum, 2003). Sebuah *app domain* tidak dapat berkomunikasi dengan *app domain* lain secara langsung. *Class* di luar *app domain* akan dianggap *remote class*.

c. Arsitektur .NET Remoting

Arsitekrut .NET Remoting dapat dilihat pada Gambar 2. Saat *client* membutuhkan *remote object*, *request* akan melalui *proxy* yang kemudian diteruskan pada *formatter*. *Formatter* ini yang mem-*package request client*, sehingga menjadi format *message* yang sesuai. Informasi kemudian dikirimkan melalui *transport channel*. *Server channel* kemudian akan menerima *message*. *Formetter* akan men-*decode message*. Nilai yang akan dikembalikan ke *client* akan melalui cara yang sama.



Gambar 2. Arsitektur .NET Remoting

Komponen *remote* tidak dapat langsung diinisiasi, sehingga diperlukan suatu aplikasi *host* untuk menginisiasi komponen tersebut. *Host* dapat berupa *Windows service*, *Windows application*, *console application*, dan lain sebagainya. Salah satu kunci dari implementasi *remoting* adalah konsentrasi penggunaan *business logic* pada satu atau beberapa *server*. Perubahan pada *business logic* tidak akan perlu untuk mengubah *coding* dari seluruh aplikasi yang terpasang, tetapi hanya perlu meng-*update* sebuah *server*. Keuntungan dari penggunaan .NET Remoting adalah kesederhanaan penggunaan pada sistem terdistribusi dengan menghasilkan performa yang optimal.

c. Infrastruktur .NET Remoting

Infrastruktur .NET Remoting membagi objek menjadi dua macam, yaitu *nonremotable* dan *remotable* (Scott McLean). Objek akan dianggap *remotable*, jika *instances* dari objek dapat melewati batasan .NET Remoting atau jika objek lain di *app domain* yang berbeda dapat mengakses *instances* tersebut. Objek *nonremotable* adalah yang sebaliknya.

Tipe objek remotable memiliki dua macam kategori, yaitu marshal-by-value (MBV) dan marshal-by-reference (MBR). *Instance* MBV dapat berada pada *app domain* yang berbeda melalui proses *serialization*. Pada MBR, objek diproses terpisah dengan mengirimkan *message* dari *proxy* pada *client* ke *server remote*. *Instance* tidak dikirimkan ke *app domain* yang lain, melainkan hanya *reference* terhadap objek yang dikirimkan.

Sebelum *instance* objek dapat diakses, maka harus dibentuk dan diinisialisasi melalui sebuah proses yang disebut objek aktivasi. MBR memiliki dua kategori aktivasi objek, yaitu *client activation* dan *server activation*. Pada *client activation*, Uniform Resource Identifier (URI) dibentuk secara dinamik di latar belakang dan dikirimkan kepada *client*.

Instance dari objek dibentuk saat *client* membentuk *proxy*. Setiap *client* mendapat *instance* dari objek.

Objek *server-activated* sering disebut juga objek *well-known* sebab *client* menggunakan URI yang sudah diketahui untuk mengakses objek. Server yang memiliki objek *remote* bertanggung jawab untuk mengkonfigurasi tipe sebagai *well-known* objek, mem-*publish* sebuah alamat (*endpoint*) *well-known*, dan mengaktifkan *instance*, jika dibutuhkan. .NET Remoting mengkategorikan *server activation* dalam dua model yaitu model Singleton dan model SingleCall.

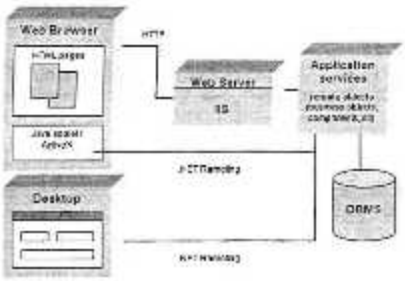
Instance pada model Singleton diaktifkan saat pertama kali diakses oleh *client* dan pada saat tidak ada *instance* lainnya. Saat aktif, *instance* Singleton menangani akses yang diminta oleh *client*. Model Singleton ini *stateful*, artinya *state* akan selalu ada. Sedangkan pada model SingleCall, *instance* dibentuk pada setiap *method* dipanggil. Objek bersifat *stateless*.

3. Perancangan Sistem

Teknologi .NET *Remoting* diimplementasikan pada suatu sistem registrasi pengambilan matakuliah. Sistem terdiri dari *business logic*, *Web interface* untuk internet, dan aplikasi *desktop* untuk intranet. Pada sistem ada pemisahan antara *business logic* dengan *database*. Diagram arsitektur sistem dapat dilihat pada Gambar 3.

Program aplikasi yang akan dibuat memiliki kelengkapan registrasi mahasiswa yang ditambah *logic* yang memeriksa tabrakan jadwal mahasiswa dan prediksi matakuliah yang diambil tiap mahasiswa berdasarkan IPK dan prasyarat matakuliah.

Objek *remote* untuk *client* akan ditempatkan pada sebuah *class* di *server*, kemudian akan diakses secara *remoting*. Objek *remote* tersebut dapat diakses tidak hanya oleh aplikasi *desktop*, tetapi juga oleh aplikasi *Web*. Pada aplikasi *Web* dibutuhkan *host*, karena itu diperlukan IIS. *Remote* objek ini yang kemudian mengakses *database*.



Gambar 3. Arsitektur Sistem

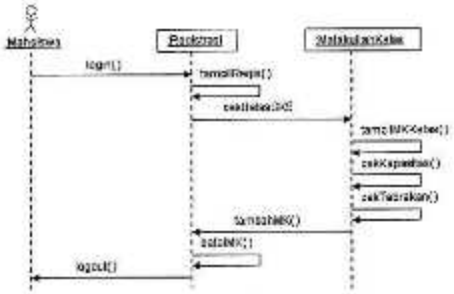
Aplikasi *client* diakses oleh banyak pengguna, yaitu mahasiswa. Untuk itu diperlukan validasi pengguna dengan cara mahasiswa memasukkan NIM dan *password*. Adanya *logic* untuk login ini berguna selain untuk keamanan juga untuk mengenali pengguna yang mengakses aplikasi.

Setelah *login* berhasil, maka mahasiswa dapat mengakses sistem registrasi. Sistem ini akan menampilkan tabel registrasi, yaitu tabel yang memuat kelas-kelas yang telah diambil oleh mahasiswa yang bersangkutan. Mahasiswa memiliki dua pilihan, yaitu untuk menambah kelas atau membatalkan kelas yang telah dia ambil. Jika mahasiswa memilih untuk menambah kelas, maka dia akan masuk pada *form* kelas yang menampilkan tabel yang memuat matakuliah yang ditawarkan. Terdapat *logic* untuk menentukan matakuliah apa yang ditawarkan ke mahasiswa yang telah *login*. Hanya matakuliah yang belum pernah diambil saja yang ditawarkan. Kemudian juga terdapat *logic* untuk memeriksa kapasitas kelas, apakah kelas sudah penuh atau

belum. Saat mahasiswa mengambil matakuliah, terdapat *logic* untuk mengecek apakah mahasiswa tersebut sudah mengambil SKS sesuai dengan batasan SKS yang berdasarkan IPK mahasiswa tersebut.

Gambar 4. Use Case Diagram Sistem

Pada Gambar 3, dapat dilihat *use case diagram* dari sistem. *Method-method* yang digunakan oleh *client* dalam melakukan registrasi mahasiswa dapat dilihat pada Gambar 4 dalam bentuk *sequence diagram*. *Method-method* yang digunakan *client* yang berbasis *desktop* maupun *client* yang berbasis *Web* adalah sama dan berada pada objek yang sama di *server*.



Gambar 5. Sequence Diagram Sistem

4. Implementasi Sistem

a. Penerapan .NET Remoting

Sistem yang dibuat terdiri dari dua bagian, yaitu sisi *server* dan sisi *client*. Pada sisi *server* terdapat *setup* data dan *output*. Sedangkan sisi *client* untuk memasukkan data registrasi. Sisi *client* ini terdiri dari dua jenis aplikasi, yaitu aplikasi *desktop* dan aplikasi *Web*. Dengan .NET Remoting, kedua aplikasi ini menggunakan *logic* yang sama dari *server*. Jika menggunakan objek *remote*, *client* maupun *server* mengakses *interface* dan objek *serializable* yang sama. Oleh karena itu dibutuhkan minimal tiga buah objek untuk mengimplementasikan .NET Remoting. Ketiga objek tersebut adalah objek yang di-*shared*, objek ini memuat objek *serializable* dan *interface*; objek *server* yang mengimplementasikan *MarshalByRefObjects*; dan objek *client* yang menggunakan keduanya.

Pada studi kasus registrasi mahasiswa, objek yang di-*shared* adalah *class* Client dan *class* IClient yang terdapat pada *project* General. Sedangkan objek *server* yang memuat

objek *serializable* adalah pada *class* ServerObject yang terdapat pada *project* Server. Semua *class client* berada di *project* Client.

b. Server

Pada Gambar 6 dapat dilihat tampilan *login* untuk masuk ke aplikasi *server*. Aplikasi server ini harus dioperasikan terlebih dahulu supaya aplikasi *client*, baik *desktop* maupun *Web* dapat dioperasikan. Jika *login* dengan benar, maka akan masuk ke *form* Menu yang dapat dilihat pada Gambar 7.

Gambar 6. Form Login Server

Form ini merupakan *form* utama *server*. Melalui *form* ini dapat diakses *setup* data mahasiswa, *setup* data dosen, *setup* data kelas, *setup* data nilai mahasiswa, serta *output* berupa Kartu Rencana Studi dan presensi kelas.

Gambar 7. Form Utama Server

c. Client

Aplikasi *client* ada dua macam yaitu aplikasi *Web* yang diakses dengan *Web browser* dan aplikasi *desktop*. Keduanya memiliki fungsi yang sama, yaitu untuk memasukkan data registrasi dari mahasiswa.

d. Aplikasi Desktop

Aplikasi *desktop* pada *client* ini terdapat tiga buah *form*, yaitu *form login* mahasiswa, *form* registrasi, dan *form* kelas yang ditawarkan. Dapat dilihat pada Gambar 8 berupa tampilan dari *form login* mahasiswa.

Gambar 8. Form Login Mahasiswa

Setelah memasukkan NIM (Nomor Induk Mahasiswa) dan *password* SSAT, bila sesuai maka akan masuk ke *form* registrasi. Jika tidak sesuai maka akan muncul peringatan. Kondisi tidak sesuai adalah bila NIM dan atau *password* tidak diisi, atau NIM dan *password* tidak sesuai dengan yang terdapat di *database*.

e. Aplikasi Web

Pada Gambar 9 dapat dilihat tampilan awal dari aplikasi *Web*. Aplikasi *Web* ini hanya dapat diakses melalui *Web browser*. Halaman pertama ini merupakan halaman *login* mahasiswa, jadi fungsinya sama seperti pada aplikasi berbasis *desktop*.

Gambar 9. Halaman Login Mahasiswa

f. Analisis Sistem

Sistem registrasi mahasiswa ini menggunakan arsitektur .NET Remoting dimana *client* menggunakan objek *remote* yang berada di *server* sebagai *logic*-nya. Agar dapat mengimplementasikan .NET Remoting, dibutuhkan sebuah objek yang di-*shared* antara *server* dan *client*. Pada studi kasus registrasi mahasiswa, objek yang di-*shared* adalah *class Client* dan *class IClient* pada *project General*. Kedua *class* tersebut di-*compile* sebagai DLL yang digunakan oleh *client* dan *server* untuk referensi.

Class IClient berisi definisi dari *interface IClient* yang akan diimplementasi oleh *server*. Sedangkan *class Client* berfungsi untuk akses ke data. Objek ini dianggap sebagai *copy* (*by value*), sehingga dibutuhkan atribut `[serializable]`. Sintaks *class Client* hanya berisi properti objek. Pada kasus ini, terdapat

banyak properti baik yang berupa *String*, *DataSet*, *boolean* dan *integer*.

Protokol yang digunakan adalah HTTP dengan membuat *channel* 1234. Selain protokol HTTP juga dapat digunakan protokol-protokol yang lain. Penggunaan HTTP karena baik *server* maupun *client* menggunakan *framework* yang sama, yaitu .NET Framework, sehingga dapat menggunakan *host* IIS. Dengan menggunakan *host* IIS, maka sistem dapat mengakses autentikasi standar yang sudah disediakan IIS.

Pada *project* Server, selain terdapat *class-class* yang berisi *method-method* dari *form* setup data, terdapat sebuah *class* yang berisi *method-method* yang digunakan oleh *client*. *Class* tersebut adalah *class* ServerObject. *Client* meminta data yang diambil dari *database*. Dari sintaks dapat dilihat data registrasi yang diambil berbentuk DataSet. Pada sisi penerima, dibuat juga DataSet baru yang kemudian diisi dari DataSet dengan format XML yang diambil dari objek *remote* di sisi *server*.

Koneksi dengan *server* terbuat dan pesan dikirimkan untuk men-*trigger method* `getClient()` pada objek Singleton di sisi *server*, yaitu *ServerObject*. *Server* kemudian membuat objek *Client* dan memenuhinya dengan data. Saat *method* dikembalikan, objek tersebut akan di-*serialized* dan semua properti *public* dan *private* akan diubah kedalam fragment XML. Dokumen XML ini di-*encapsulated* dalam bentuk SOAP dan dikembalikan ke *client*. Framework .NET Remoting di *client* men-*generate* objek Client yang baru dan mengisinya dengan data yang diterima dari sisi *server*. *Client* memiliki *copy* yang sama persis dengan objek Client yang dibuat di *server*.

Hal yang sama juga terjadi di *client* berbasis *Web*. *Client* ini juga memanggil *class* yang berada di sisi *server* dengan cara yang sama. Oleh karena objek di *server* yang diakses oleh kedua aplikasi dengan basis yang berbeda tersebut adalah sama, maka akan memudahkan bila ada *logic* yang perlu diganti. Aplikasi *client*, baik yang berbasis *desktop* maupun yang berbasis *Web* hanya berisi sintaks untuk tampilan (*user interface*) dan sintaks untuk memanggil *method-method* yang berada di sisi *server*. Bila terjadi perubahan *logic* registrasi, yang perlu diubah hanya pada sisi *server*.

5. Kesimpulan

Teknologi .NET Remoting memungkinkan digunakannya objek *remote* yang dipanggil oleh *client* dengan basis yang berbeda. Seluruh *logic* yang kompleks dapat diletakkan pada sisi *server*, sehingga *client* hanya berisi sintaks tampilan dan sintaks untuk memanggil *methods* yang berada di sisi *server*. Teknologi tersebut memudahkan dalam hal perubahan pada *logic*. Perubahan *logic* cukup dilakukan sekali pada sisi *server*, sehingga tidak perlu dilakukan instalasi ulang pada sisi *client*.

Sistem *client* dan *server* harus menggunakan .NET Framework yang sama, sehingga penggunaan protokol HTTP menjadi lebih efektif saat komponen *remote* menggunakan *host* IIS sebab autentikasi standar sudah disediakan IIS.

Implementasi teknologi .NET Remoting pada studi kasus registrasi mahasiswa UKDW telah berhasil dilakukan. Sistem registrasi mahasiswa UKDW yang terdiri dari *client* dengan aplikasi berbasis *desktop* dan *Web* telah dapat berjalan dengan menggunakan *logic* yang sama yang diambil dari objek *remote* di sisi *server*.

6. Saran

Pendistribusian *logic* dan *database* pada sistem dapat diletakkan pada lebih dari satu *server* sehingga kemungkinan kerja sistem dapat lebih optimal. Serta penelitian lebih untuk melihat jumlah *client* maksimal yang dapat mengakses objek *remote* pada waktu yang bersamaan.

Daftar Pustaka

Blum, Richard, 2003, *C# Network Programming*. Sybex.

MacDonald, Matthew, 2003, *Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting*. Microsoft Press.

McLean, Scott dan James Naftel dan Kim, 2002, Williams. *Microsoft .NET Remoting*. Microsoft Press.

Rammer, Ingo, 2004, *Advanced .NET Remoting*. Apress.

Sommerville, Ian, 2001, *Software Engineering* (6th Edition), Prentice Hall

Völter, Markus dan Michael Kircher dan Uwe Zdun, 2005, *Remoting Patterns*. West Sussex: John Wiley & Sons Ltd.