ANALISIS IMPLEMENTASI *RESTFULL* WEB SERVICE MENGGUNAKAN *RESOURCE-ORIENTED ARCHITECTURE*

Meyti Eka Apriyani[1], Elok Nur Hamdana[2]

[1,2] Jurusan Teknologi Informasi, Politeknik Negeri Malang
[1]meyti24@gmail.com, [2] elokhamdana@gmail.com

Abstrak

Web *service* merupakan kunci integritas untuk aplikasi yang berbeda *platform*, bahasa, dan sistem. *Representational State Transfer* (REST) merupakan prinsip aplikasi web *service* dengan menambahkan sebuah elemen untuk menggunakan *Uniform Resource Identifier* (URI) standar, memberikan kepentingan kepada penggunaan HTTP serta menjadi metode yang lebih baik untuk membangun web *service*. Filosofi REST mendukung prinsip dan *protocol* yang ada di web untuk membuat web *service* yang lebih *powerful*. *Resource-Oriented Architecture* (ROA) menetapkan sejumlah batasan dalam pembangunan aplikasi *RESTful* web *service*, yaitu addressability, statelessnes, uniform interfaces, representations, dan connectedness. Pemanfaatan web service dengan metode ROA pada Perusahaan dengan skala besar dapat memenuhi kebutuhan bisnis dan dapat membangun sistem yang memiliki keuntungan dalam skalabilitas interaksi komponen, *generality of interface*, komponen yang berdiri sendiri dan komponen penengah yang dapat mengurangi *latency*, menekankan keamanan, dan mengenkapsulasi *legacy-system*. Analisis dan implementasi *RESTful* Web *Service* terhadap aplikasi berupa kinerja dan trafik data manajemen menggunakan *Resource-oriented Architecture* agar aplikasi yang menggunakan web *service* lebih stabil dan web *service* membantu pengiriman data dari dan ke database berupa antrian eksekusi yang teratur.

Kata kunci: *HTTP, Resource-oriented Architecture, REST, URI, Web Service*

1. Pendahuluan

Perusahaan dengan skala besar yang memiliki beberapa sistem informasi berbasis web, *desktop* maupun *mobile* dengan spesifikasi berbeda memiliki sebuah sarana untuk mengintegrasikan sistem tersebut sehingga pertukaran data antar sistem dapat dilakukan dengan baik. Perusahaan tersebut memanfaatkan sebuah teknologi web *service* yang mampu menjembatani aplikasi yang berbeda bahasa pemrogramannya maupun sistem operasinya.

Web *service* merupakan kunci integritas untuk aplikasi yang berbeda *platform*, bahasa, dan sistem. Dengan kata lain dapat menyebut web *service* sebagai "titik temu bisnis". *Representational State Transfer* (REST) berupa kumpulan prinsip aplikasi web *service* dengan menambahkan sebuah elemen untuk menggunakan *Uniform Resource Identifier* (URI) standar, memberikan kepentingan kepada penggunaan HTTP serta menjadi metode yang lebih baik untuk membangun web *service*.

Resource-Oriented Architecture (ROA) menetapkan sejumlah batasan dalam pembangunan aplikasi *RESTful web service*, yaitu *addressability*, *statelessnes*, *uniform interfaces*, *representations*, dan *connectedness*, yang jika diterapkan secara menyeluruh akan menghasilkan sistem yang memiliki keuntungan dalam skalabilitas interaksi komponen, *generality of interface*, komponen yang berdiri sendiri, dan komponen penengah yang dapat mengurangi *latency*, menekankan keamanan, dan mengenkapsulasi *legacy-system.*

Filosofi REST mendukung prinsip dan protocol yang sudah ada di web cukup untuk membuat web service yang kuat (*robust*). Hal ini berarti bahwa developer yang mengerti HTTP dan XML dapat mulai membangun web *service* tanpa membutuhkan *toolkit* di belakang apa yang biasanya digunakan dalam pengembangan aplikasi internet.

2. Tinjauan Pustaka

2.1 Kajian Terdahulu

Wilson A. Higashino dkk (2009) melakukan penelitian mengenai penggunaan REST pada web service. Penelitian tersebut dimuat dalam *Proceedings First International Symposium on Services Science*. Penelitian tersebut mendapat hasil sebagai berikut:

1. Artikel merepresentasikan konsep dan karakteristik dari RESTful web *service*, terlihat pada beberapa area dimana banyak standar yang telah diaplikasikan ke dalam web *service*.
2. Aspek terpenting dalam RESTful web *service* adalah *QoS*, transaksi, keamanan, dan pesan. Dalam artikel REST didefinisikan sebagai *style* arsitektur untuk mendefinisikan sistem *hypermedia* seperti web.

3. REST dapat diimplementasikan pada web *service*, serta dapat digunakan untuk karakteristik dari layanan web.
4. Berdasarkan hasil penelitian tersebut, sangat penting menggunakan arsitektur *REST* dan *ROA* ke dalam *web service*. Maka penelitian ini akan mencoba membangun simulasi *web service* menggunakan *ROA* dan studi kasus pada perusahaan berskala besar

2.2 Landasan Teori

1. Web Service

Menurut Robert Daigneau (Daigneau, 2012:6) alasan menggunakan *web service* adalah kemudahan dalam penggunaan kembali (*reuse*) dan berbagi (*share*) logika yang sama dengan klien yang beragam seperti *mobile*, desktop, dan aplikasi web. Jangkauan *web service* yang luas karena *web service* bergantung pada standar yang terbuka, dapat beroperasi pada *plaftorm* yang berbeda, serta tidak bergantung pada teknologi eksekusi yang mendasarinya. Semua *web service* setidaknya menggunakan HTTP dan format pertukaran data standar berupa XML, JSON, atau media lain. Selain itu, web service menggunakan HTTP dalam dua cara yang berbeda yaitu sebagai protokol standar untuk menentukan perilaku standar pelayanan serta sebagai media transportasi untuk menyampaikan data.

2. REST Web Service

REST merupakan suatu arsitektur perangkat lunak untuk sistem hypermedia yang terdistribusi. Contoh sistem yang mengimplementasikan REST adalah *world wide web* (www). REST lebih kepada filosofi lama, ketimbang sebuah teknologi yang baru. Tetapi dalam kenyataannya datang kemudian dalam teknologi. REST *web service* terdiri dari dua istilah yaitu "REST" dan "*web service*". Menurut Roy Thomas Fielding, REST (*Representational State Transfer*) adalah model arsitektur yang pada dasarnya memanfaatkan teknologi dan protokol yang sudah ada seperti HTTP (*Hypertext Transfer Protocol*) dan XML. REST web service merupakan web service yang dibangun dengan memanfaatkan teknologi dan protokol yang sudah ada. Desain REST web service lebih mirip seperti pekerjaan seni daripada sains, hal ini dikarenakan desain REST harus dapat menjawab permasalahan yang dihadapi.

3. ROA

ROA didasarkan pada konsep *resource* yang masing-masing berupa komponen yang didistribusikan langsung dan setiap *resource* secara unik diakses dengan menggunakan URI, setiap resource menggunakan antarmuka yang sama untuk pengiriman state antara klien dan *resource*, menggunakan protokol yang mendukung *client-server*, *stateless*, *cacheable*, dan *layered*.

4. IIS7

IIS adalah sebuah *role built-in* di Windows *Server* 2008 R2 (dan tentunya sudah ada sejak Windows XP), yang berfungsi mengubah *server* menjadi web server. Web server berbasis IIS7 sudah mendukung *web programming* berbasis ASP dan ASP.Net dengan *DotNet Framework* 3.51 SP1 dan dapat diupgrade ke *DotNet Framework* 4. Selain itu IIS7 dapat diinstal pada *server* yang sudah diaktifkan role AD maupun belum. IIS juga dapat membangun *hosting* dengan berbagai *web site* yang dihubungkan dengan *network connection* yang berbeda-beda, termasuk koneksi SSL/HTTPS serta dapat melayani fungsi Web *Distributed Authoring and Versioning* (WebDAV).

3. Perancangan

3.1 Objek Penelitian

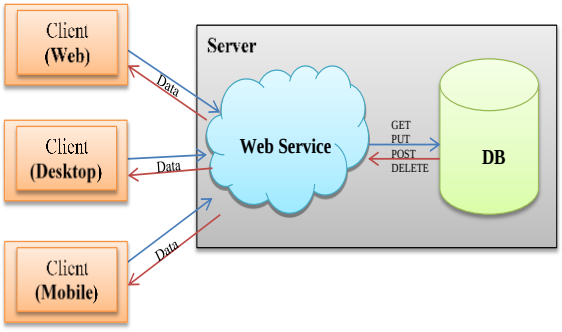
Penelitian ini dilakukan pada perusahaan berskala besar. Objek penelitian adalah simulasi RESTful web service menggunakan *Resource-oriented Architecture* (ROA), sesuai kebutuhan *services* pada perusahaan tersebut.

3.2 Metodologi Penelitian

Penelitian ini mengimplementasikan metode *Resource-oriented Architecture* (ROA) pada perusahaan. Penelitian ini berusaha membuat simulasi web service berdasarkan analisa yang telah dilakukan pada perusahaan tersebut. Tidak *over load* tapi tetap efisien dalam penggunaan sumberdaya perangkat lunak.

Pengambilan data penelitian menggunakan metode percobaan laboratorium. Percobaan untuk memperoleh data primer berupa hasil pengujian berdasarkan variabel yang telah ditentukan. Selain itu penelitian kepustakaan untuk memperoleh data sekunder yang mendukung penelitian ini seperti jurnal dan buku sebagai bahan rujukan.

3.3 Desain Sistem



Gambar 1 Desain Sistem

Langkah dalam mendesain sistem adalah sebagai berikut:

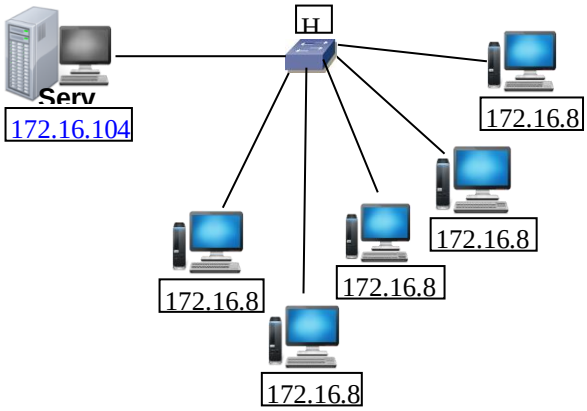
1. Perancangan RESTful *Web Service*
   Perancangan sistem RESTful *web service* menggunakan metode *Resource-oriented Architecture* (ROA) berdasarkan analisis

pemanfaatan sistem RESTful web service pada perusahaan berskala besar.

2. Simulasi RESTful *Web Service*
   Merancang simulasi RESTful *web service* berdasarkan analisis pemanfaatan RESTful dengan metode *Resource-oriented Architecture* (ROA) yang telah dilakukan. Sehingga mendapatkan deksripsi arsitektural dari RESTful itu sendiri.
3. Pengujian *RESTful Web Service*
   Menguji simulasi sistem RESTful *web service* yang telah dibuat untuk fungsional *web service* yang diterapkan pada perusahaan berskala besar.
4. Analisis simulasi *RESTful Web Service*
   Menganalisis sistem RESTful berdasarkan simulasi yang telah teruji menggunakan metode *Resource-oriented Architecture* (ROA)

3.4 Topologi Jaringan

Komputer yang mengakses berada dalam satu jaringan LAN dengan server. Topologi yang digunakan adalah topologi star. Adapun topologi jaringan pada simulasi RESTful web service, digambarkan pada gambar 2 sebagai berikut:



Gambar 2 Topologi Jaringan Simulasi Sistem

Terdapat lima komputer klien untuk mengakses web simulasi RESTful web service, memberikan perintah query yang disampaikan pada web service sehingga dapat dieksekusi database. Dengan pengujian lima komputer klien, tracebility pada web aplikasi, web service dan database lebih mudah dengan pencapaian hasil pengujian lebih akurat.

4. Hasil dan Pengujian

4.1 Skenario Pengujian

Sebelum membahas hasil pengujian, perlu adanya penjabaran secara detail skenario 4. pengujian simulasi RESTful web service yang dilakukan. Pengujian dilakukan pada aplikasi web pendukung metode-metode ROA (GET, POST, PUT, DELETE). Aplikasi mengirim sebuah perintah yang akan dieksekusi oleh web service berupa logic-logic string untuk disampaikan kepada database. Tool yang digunakan dalam pengujian ini adalah Activity Monitoring dari MS SQL Management Studio untuk mengukur antrian eksekusi perintah yang diminta oleh aplikasi web per user akses.

Pengujian dilakukan secara bersamaan untuk mengetahui antrian eksekusi pada database. Setiap client mengakses aplikasi web tersebut, pada sisi server dilakukan monitoring eksekusi perintah dari klien. Setiap klien akan dicatat dalam melakukan eksekusi yang ditunjukkan dengan tampilan grafik, jika web service mengeksekusi perintah tidak membutuhkan waktu lama (timeout) atau melakukan query yang membutuhkan cost performace yang tinggi (expensive query) maka pencatatatan pada moinitoring akan mencatat dengan kondisi penimpaan (replace) dan jika terjadi antrian lama, membutuhkan waktu lama (timeout) serta membutuhkan cost performance tinggi (expensive query) maka proses tersebut akan di kill proses hingga tidak menyebabkan terjadinya down pada sisi server.

4.2 Analisis

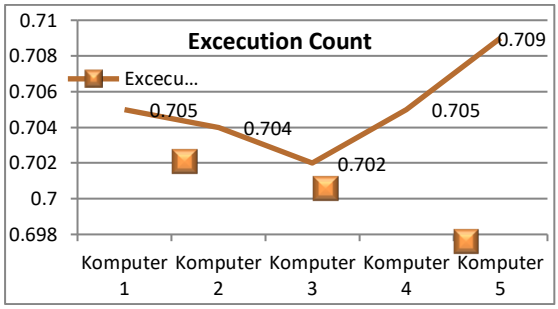
Analisis Implementasi RESTful web service menggunakan metode Resource-Oriented Architecture (ROA) dilakukan setelah mendapat respon dari masing-masing logic-logic dalam metode ROA yang dijalankan pada web service yaitu POST, PUT, GET dan DELETE.

Waktu eksekusi yang diuji oleh perintah (*query*) mendapatkan hasil eksekusi *count* setiap ms (*milli second*) per komputer pada tabel dibawah ini:

Tabel 1 Eksekusi Database

No	Komputer Tes	Excecution Count (s)
1	Komputer 1	0.705
2	Komputer 2	0.704
3	Komputer 3	0.702
4	Komputer 4	0.705
5	Komputer 5	0.709

Proses eksekusi perintah menunjukkan rata-rata *user* akses eksekusi *count* jika dihitung tidak mencapai satu detik yang menunjukkan data eksekusi normal ketika proses *login user* ke *database* yang menampilkan proses pemanggilan data yang berjalan berupa *authentication user* ke *active directory* tidak berlangsung lama, hasil tabel diatas berupa grafik dapat dilihat pada gambar 3 Grafik eksekusi *count* dibawah ini:



Gambar 3 Grafik Eksekusi *Count*

Berdasarkan simulasi aplikasi web menggunakan RESTful *web service* tentunya berorientasi pada kinerja dan trafik *management* data, pengaturan trafik yang diberikan kepada server oleh *web service* sangat efisien dalam proses eksekusi perintah ke *database*. Terlihat dari grafik *eksekusi count* pengiriman perintah *login* oleh lima *user* akses dari *web service* ke *database* kurang dari satu menit sesuai dengan waktu *default time out* eksekusi dari *web service*. Apabila melebihi dari satu menit dapat menyebabkan terjadinya *time out*

5. Kesimpulan dan saran

5.1 Kesimpulan

Berdasarkan tahapan pengujian simulasi web dengan RESTful web service dan analisa terhadap database yang telah dilakukan, dapat ditarik kesimpulan:

1. Simulasi yang dibangun menggunakan metode Resource-oriented Architecture (ROA), pada studi kasus menggunakan lima komputer dengan satu server yang dipantau dengan tool Activity Monitoring sehingga dapat dilihat hasil eksekusi setiap per detiknya.
2. Analisa terhadap aplikasi berupa kinerja dan trafik data manajemen menggunakan Resource-oriented Architecture agar aplikasi data multimedia yang menggunakan web service lebih stabil dan web service membantu pengiriman data dari dan ke database berupa antrian eksekusi yang teratur.
3. Pemakaian perintah query pada aplikasi menggunakan DDL dan DML yang dirangkum ke dalam store procedure di database.

5.2 Saran

Dari tahapan pengujian simulasi web aplikasi dengan web service dan analisa terhadap database yang telah dilakukan, saran-saran untuk melanjutkan analisis ini adalah:

1. Melakukan pemisahan antara server database dengan server web service.
2. Melakukan penelitian mengenai user akses terhadap database.
3. Menggunakan database selain MS SQL seperti oracle atau MySQL.
4. Melakukan penelitian dengan menggunakan data multimedia kompleks.
5. Melakukan pengujian berbasis cloud computing pada jaringan internet.

Daftar Pustaka:

[1] A. Higashino, Wilson et al, 2009, "REST and Resource-Oriented Architecture" Proccedings First International Symposium on Service Science ISSS'09, Logos, Berlin tersedia:http://isss.uni-leipzig.de/index.php/Download-document/76paper_higashino .html diakses pada: 23 Oktober 2014

[2] Daigneau, Robert. "Service Design Patterns: Fundamental Design Solution for SOAP/WSDL and RESTful Web Services", tersedia: http://repository.amikom.ac.id/files/Publikasi_10.11.3910.pdf diakses pada: 16 Oktober 2014

[3] Dudhe, Anil, "Performance Analysis of SOAP and RESTful Mobile Web Service in Cloud Environment", International Journal of Computer Application, India 2014, Tersedia: http://research.ijcaonline.org/rtinfosec/number1/rtinfosec1401.pdf, diakses pada 27 Oktober 2014

[4] Lucky. 2008. "XML Web Service Aplikasi Desktop, Internet, & Handphone", Jakarta: Penerbit Jasakom

[5] L. Costello, Roger, 2002, "REST (Representational State Transfer)", tersedia: http://www.xfront.com/sld001.htm, diakses pada 23 Oktober 2014

[6] Novendriono, Yohanes. "Pengembangan Aplikasi RESTful Web Service Menggunakan Resource-Oriented Architecture", Makalah Tugas Akhir Universitas Atma Jaya, Yogyakarta 2011

[7] Oracle and/or its Affiliates, 2013," What Are RESTful Web Services?", tersedia: http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html, diakses pada 23 Oktober 2014.

[8] Riyadi, Damar. "Rancang Bangun Web Service Untuk Perbandingan Harga Pengiriman Dengan Metode Web Scrapping Dan Pemanfaatan API", Naskah Publikasi Sekolah Tinggi Manajemen Informatika Dan Komputer Amikom, Yogyakarta 2013