RANCANG BANGUN PERANGKAT LUNAK AGENT UNTUK POLLING HALAMAN WEB PADA SELANCAR WEB DALAM KOMUNITAS MENGGUNAKAN AGLET

Intan Yuniar Purbasari [1], Joko Lianto Buliali[2]

[1] *Teknik Informatika, FTI-UPN "Veteran" Jawa Timur email: intan@upnjatim.ac.id*
[2] *Teknik Informatika FTIF-ITS, Surabaya*

***Abstract**, While surfing the internet, it's common to browse a page which has not been visited before. If the user finds the page interesting, probably the links within the page will be visited too, in order to find more relevant information. Following links of an unknown websites takes more time and may lead to irrelevant information. On the other side, there might be some users with the same interest, have been visited the same page and have found out which links are useful and which are not. This application records useful links selected by users and will be shown to other users with the same community of interest. Aglet is used to connect client application to a database for its autonomy, its responsive ability towards its execution environments, and its ability to communicate with each other. This experiment has been conducted in an intranet environment. The result showed that a user could get some recommendation by other users on a web addresses and they could also participate in giving recommendation.*

Keywords: Browsing, Software Agent, Recommendation, Interest, Community, Aglet.

Selancar Internet merupakan kegiatan yang menyenangkan bagi kebanyakan pengguna Internet. Aktivitas ini merupakan salah satu aktivitas Internet yang paling populer [7]. Kegiatan yang selama ini dilakukan secara individual memiliki efisiensi jelajah Web yang rendah seperti yang akan digambarkan berikut.

Apabila seseorang berada pada suatu halaman Web yang belum pernah dikunjunginya sebelumnya dan menemukan bahwa halaman Web tersebut menarik baginya serta ada sejumlah *link* dari halaman tersebut, maka besar kemungkinan yang bersangkutan akan mencoba mengunjungi *link-link* tersebut dengan harapan akan dapat menemukan informasi tambahan lain yang relevan. Mengikuti *link-link* dari suatu halaman Web yang belum dikenal tidak berbeda dengan kegiatan trial and error.

Ini membutuhkan energi dan waktu yang tidak sedikit. Di lain pihak, sangat mungkin individu-individu lain yang memiliki *interest* yang sejenis dengan peselancar Internet tersebut telah pernah melakukan penelusuran *link-link* yang sama dari halaman Web yang sama sebelumnya, serta telah menemukan *link* mana yang bermanfaat besar dan mana yang kurang bermanfaat baginya.

Apabila pengalaman individu-individu lain tersebut dalam penelusuran *link-link* dari halaman Web yang sama dapat di-*share*, sangat mungkin akan meningkatkan efisiensi jelajah Web individu yang baru pertama kali mengunjungi halaman Web tersebut.

Software *agent* yang akan dirancang pada aplikasi ini dapat mencatat *link-link* dari suatu halaman Web yang oleh seseorang (dalam suatu komunitas peselancar Internet) dianggap bermanfaat dan kurang bermanfaat. Semakin banyak individu yang memiliki minat yang sama berpartisipasi dalam penentuan bermanfaat tidaknya *link-link* dari suatu halaman Web, semakin representatif kebenaran informasi ini. Semakin aktif kelompok tersebut menjelajahi halaman-halaman Web dan memberikan pendapatnya, semakin representatif pula rekomendasi yang diberikan *agent*.

Tujuan dari pembuatan aplikasi ini adalah membuat suatu perangkat lunak *agent* dengan menggunakan aglet sebagai pembawa pesan di dunia Internet. Dengan digunakannya aglet pada perangkat lunak *Agent*, diharapkan efektivitas selancar Web dalam komunitas dapat ditingkatkan. Manfaat dari pembuatan aplikasi ini adalah : Pengguna yang baru mengunjungi sebuah halaman web untuk pertama kali akan mendapat informasi mengenai link mana

yang bermanfaat, para pengguna dalam satu komunitas interest yang sama akan dapat saling bekerja sama untuk meningkatkan efektivitas selancar web.

PERANGKAT LUNAK AGENT

Pengertian

Perangkat lunak *agent* adalah program yang membantu manusia dalam melakukan pekerjaannya dan bertindak sesuai keinginan mereka [4]. Penjelasan tersebut dilihat dari sudut pandang *end-user*.

Jika dilihat dari sudut pandang sistem, perangkat lunak *agent* adalah obyek perangkat lunak yang:

- berada dalam lingkungan eksekusi
- memiliki karakteristik wajib sebagai berikut:
  - reaktif: merasakan perubahan dalam lingkungan eksekusinya dan bertindak sesuai perubahan tersebut
  - autonomous: memiliki kontrol atas segala tindakannya
  - memiliki tujuan pasti: selalu proaktif
  - berkelanjutan: selalu dieksekusi secara kontinu
- dan dapat memiliki karakteristik opsional sebagai berikut:
  - komunikatif: dapat berkomunikasi dengan *agent* lain
  - bergerak *(mobile)*: dapat bergerak dari satu *host* ke *host* lain
  - belajar: beradaptasi berdasarkan pengalaman terdahulu
  - dapat dipercaya: dapat dipercaya oleh *end-user*.

Mobilitas adalah salah satu karakteristik opsional perangkat lunak *agent*, yang artinya tidak semua *agent* dapat bergerak *(mobile)*. *Agent* yang dapat bergerak disebut *mobile agent,* sedangkan *agent* yang tidak bergerak disebut *stationary agent*.

Stationary Agent

Stationary agent adalah *agent* yang hanya dieksekusi di mana ia pertama kali dieksekusi [4]. Jika *agent* membutuhkan informasi yang tidak ada pada sistem tersebut atau jika ia butuh untuk berinteraksi dengan *agent* pada sistem lain, maka *agent* menggunakan mekanisme komunikasi seperti *Remote Procedure Calling* (RPC).

Contoh aplikasi *stationary agent* adalah model *store-and-forward* pada *client* POP *(Post Office Protocol)* yang berkomunikasi dengan *server* SMTP (*Simple Mail Transfer Protocol*) [9].

Mobile Agent

Mobile agent adalah *agent* yang tidak terikat pada sistem di mana ia mulai dieksekusi [4]. *Mobile agent* memiliki kemampuan unik untuk memindahkan dirinya dari satu sistem ke sistem lain pada suatu jaringan. Kemampuan untuk bergerak memungkinkan *mobile agent* dapat berpindah menuju sistem yang memiliki obyek yang diinginkan *agent* untuk berinteraksi, sehingga *agent* dapat mengambil keuntungan karena berada dalam *host* atau jaringan yang sama dengan obyek tersebut.

Aglet

Aglet adalah obyek *mobile* Java yang dapat mengunjungi *host* lain di jaringan komputer [1]. *Host* yang dikunjungi harus berupa *host* yang memungkinkan untuk menerima aglet. Ketika aglet berpindah, ia membawa serta kode program dan *state* dari semua obyek yang dibawanya. Aglet juga bersifat autonomous, karena berjalan pada *thread* eksekusi tersendiri setelah tiba di suatu *host*, dan bersifat reaktif, karena dapat merespon pesan yang datang.

Proxy adalah perwakilan dari aglet. Ia bertindak sebagai perisai yang melindungi aglet dari akses langsung ke *method* publiknya. Proxy juga dapat "menyembunyikan" lokasi aglet sebenarnya. Artinya aglet dan proxynya dapat berada pada *host* yang berbeda, sehingga proxy (yang selalu lokal) dapat menyembunyikan keberadaan aglet. *Proxy* dari aglet adalah nilai kembalian pada proses *creation* dan *cloning*.

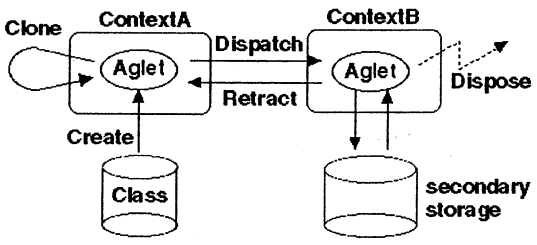
Context adalah tempat kerja (*workplace*) aglet. Context merupakan obyek stasioner yang menyediakan sarana untuk mengatur aglet yang sedang aktif dalam lingkungan eksekusi uniform. Satu *node* pada jaringan komputer dapat menjalankan beberapa proses *server* (*engine*) dan tiap *server* dapat memiliki beberapa context. Context dapat dikenali dari kombinasi alamat *server* dan nama *context* itu sendiri.

Identifier adalah identitas unik yang dimiliki oleh setiap aglet. Identitas ini tidak berubah selama waktu hidup (*lifetime*) aglet.

Operasi Dasar Aglet

Beberapa operasi dasar yang dapat dilakukan pada aglet adalah sebagai berikut:

1. *Creation* (Pembuatan). Pembuatan aglet terjadi di dalam *context*. Aglet baru diberi *identifier*, dimasukkan ke dalam *context*, dan diinisialisasi. Aglet langsung dieksekusi setelah sukses diinisialisasi.
2. *Cloning* (Pengklonan). Pengklonan aglet menghasilkan aglet yang hampir sama dengan aglet asal pada *context* yang sama. Perbedaannya hanya pada *identifier* yang diberikan pada aglet baru dan eksekusi dimulai pada aglet baru. Yang perlu diperhatikan adalah bahwa *thread* eksekusi tidak ikut diklon.
3. *Dispatching* (Pengiriman). Pengiriman aglet dari satu *context* ke *context* lain akan memindahkan aglet dari *context* asalnya dan dimasukkan ke *context* baru, dimana ia me-*restart* eksekusinya (*thread* eksekusi tidak ikut berpindah). Dapat dikatakan bahwa aglet "didorong" ke *context* baru.
4. *Retraction* (Penarikan). Penarikan aglet akan menarik/memindahkannya dari *context* sekarang dan memasukkannya ke *context* di mana proses penarikan *(retraction)* dilakukan.
5. *Activation* dan *deactivation* (pengaktifan dan penon-aktifan). Penon-aktifan adalah kemampuan aglet untuk menghentikan eksekusinya untuk sementara dan menyimpan *state* ke penyimpanan sekunder. Aktifasi aglet akan "menghidupkan" aglet kembali di *context* yang sama.
6. *Disposal* (Pembuangan). Pembuangan aglet akan menghentikan eksekusi aglet dan memindahkannya dari *context*-nya.



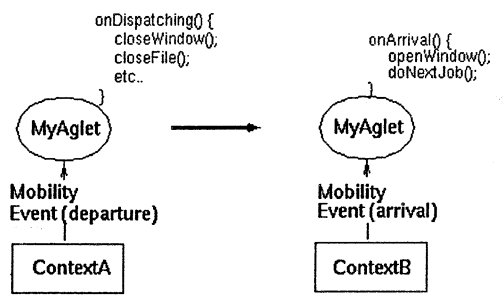
Gambar 1. Model Siklus Hidup Aglet

Model Event Aglet

Model pemrograman aglet berbasiskan *event*. Model ini memungkinkan pemrogram untuk menyisipkan *listener* tambahan ke dalam aglet. *Listener* menangkap *event* tertentu dalam siklus hidup aglet sehingga *action* tertentu dapat dilakukan ketika sebuah *event* terjadi.

Tiga macam *listener* adalah sebagai berikut:

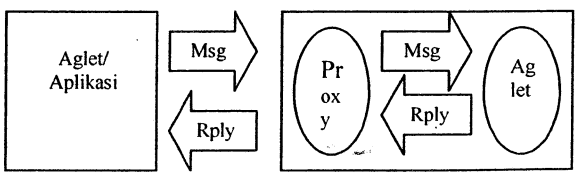
1. *Clone listener*. *Listener* ini digunakan untuk memantau *event* pengklonan. *Action* yang dapat dilakukan dengan *listener* ini adalah ketika aglet akan diklon, ketika klon sudah dibuat, dan setelah proses klon dilakukan.
2. *Mobility listener*. *Listener* ini digunakan untuk memantau *event* mobilitas. *Action* yang dapat dilakukan dengan *listener* ini adalah ketika aglet akan dikirim ke *context* lain, ketika akan ditarik dari *context* lain, dan ketika aglet tiba di *context* baru.
3. *Persistence listener*. *Listener* ini digunakan untuk memantau *event* persisten. *Action* yang dapat dilakukan dengan *listener* ini adalah ketika aglet akan dinonaktifkan dan ketika aglet sudah diaktifkan.



Gambar 2. Contoh Model Event Aglet

Model Komunikasi Aglet

Aglet berkomunikasi dengan menggunakan *message passing* [6]. Fasilitas ini memungkinkan aglet untuk membuat dan saling bertukar pesan dengan fleksibel.



Gambar 3. Komunikasi Antar-Aglet

Pesan pada aglet berupa obyek. Obyek pesan dikenali dari jenisnya (*kind*). Properti string ini digunakan untuk membedakan antara pesan satu dengan lainnya. Obyek pesan juga dapat berisi argumen opsional untuk data yang berhubungan dengan pesan tertentu. Argumen

ini dapat berupa argumen atomik (bertipe `String`, `int`, dan sebagainya) atau tabular (bertipe `Hashtable`).

Untuk dapat menangani pesan yang datang, aglet memiliki *method* `handle Message (Message msg)`.

Aglet juga dapat melakukan komunikasi secara *remote* (*remote message passing*). *Remote messaging* dapat dilakukan antara proxy aglet (lokal) dan aglet (kemungkinan *remote*). Argumen yang dimiliki oleh pesan yang akan dikirim secara *remote* harus berupa argumen yang dapat diserialisasi, atau dengan kata lain mengimplementasikan *java.io.Serializable*. *Remote messaging* tidak menyebabkan transfer kode byte, oleh karena itu *class-class* yang digunakan pada pengiriman pesan harus tersedia di kedua *host*.

Arsitektur Sistem Aglet

Arsitektur aglet terdiri atas dua lapisan (*layer*) dan dua API yang mendefinisikan *interface* untuk mengakses fungsi-fungsinya.

Aglet Runtime Layer

Layer ini juga terdiri atas kerangka kerja inti (*core framework*) dan sub-komponen. Kerangka kerja inti menyediakan mekanisme dasar untuk eksekusi aglet berikut ini:

- Serialisasi dan deserialisasi aglet
- *Class loading* dan transfer
- *Reference Management* dan pengumpulan sampah *(garbage collection)*

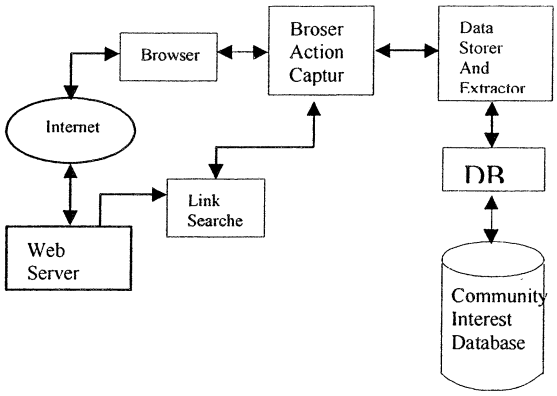
Communication Layer

Aglet menggunakan *Agent* Transfer Protocol (ATP) sebagai implementasi default dari layer komunikasi. ATP dibuat berdasarkan protokol HTTP dan merupakan protokol level aplikasi untuk transmisi *mobile agent*.

METODOLOGI

Gambaran Umum Sistem

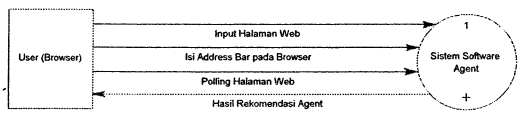
Perangkat lunak *agent* terdiri atas dua bagian, yaitu *client* dan *server*. Gambaran umum perangkat lunak *agent* digambarkan sebagai berikut :



Gambar 4. Gambaran Umum Sistem

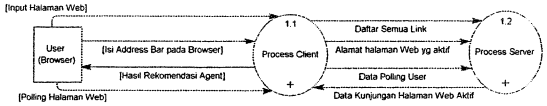
Data Flow Diagram Sistem

Proses serta alur data pada sistem ini dapat dilihat pada data flow diagram (DFD) berikut ini :



Gambar 5. DFD LEVEL 0

Data flow diagram ini kemudian di *breakdown* menjadi DFD level 1 berikut :



Gambar 6. DFD LEVEL 1

Algoritma Mode Learn dan Mode Use

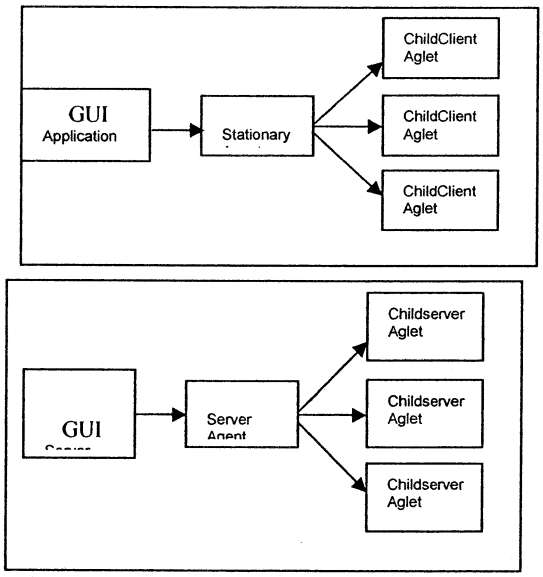
Yang dimaksud dengan *Use* Mode adalah modus operasi "bukan Learn". Penamaan modus operasi penggunaan (Use mode) ini sebenarnya kurang menggambarkan maksud sebenarnya, karena dalam kenyataannya dalam penggunaan *agent*, *Use* mode dan *Learn* mode berlangsung pada saat yang bersamaan.

Pada Use mode ini, pengguna hanya akan sekedar menggunakan rekomendasi dari *agent* saja. Rekomendasi diurutkan berdasarkan popularitas, yaitu rasio banyaknya individu yang menyukai suatu *link* dan frekuensi *link* tersebut dikunjungi individu dalam kelompok. Popularitas dihitung dengan rumus sebagai berikut :

$$popularitas = \frac{banyaknya\_individu\_yang\_menyukai\_suatu\_link}{frekuensi\_kunjungan\_link}$$

Arsitektur Sistem

Arsitektur sistem perangkat lunak *agent* ini, meliputi arsitektur *client* dan *server*, dapat digambarkan seperti pada berikut ini:



Gambar 7. Arsitektur Client Dan Server

PEMBAHASAN DAN UJI COBA

Implementasi Basis data

Implementasi basis data untuk aplikasi server terdiri dari dua tabel yang diperlukan untuk menyimpan data halaman Web yang pernah dikunjungi oleh pengguna yaitu:

- Tsource
  Tabel ini menyimpan data URL yang pernah dikunjungi oleh pengguna.
- Tlink
  Tabel ini menyimpan semua *link* dari tiap *Source* yang ada pada tabel TSource.

Implementasi Proses Server

Pada tahap ini diimplementasikan *class-class* yang diperlukan untuk membuat aplikasi yang dapat men-*set up server* aglet untuk menerima *request* dari *client.*

Implementasi Antarmuka

Pengimplementasian antarmuka untuk aplikasi *server* hanya berhubungan dengan proses *start* dan *shutdown* context aglet serta *create* dan *dispose Server*Aglet.

Implementasi Proses Client

Pada tahap ini diimplementasikan *class-class* yang diperlukan untuk membuat aplikasi yang dapat men-*set up context* dan fungsi *server* pada sisi *client* serta menampilkan data halaman Web.

Implementasi Antarmuka

Pengimplementasian antar muka dilakukan sesuai dengan kebutuhan proses *client* dalam menjalankan fungsi *server*, memanggil *browser* Internet Explorer, menampilkan data hasil rekomendasi, dan melakukan *polling* kepada pengguna.

Lingkungan Ujicoba

Komputer yang digunakan sebagai *server* adalah komputer LP-39 dengan IP Address 10.126.11.139. Spesifikasi hardware yang digunakan pada komputer ini adalah Pentium III Celeron dengan memori fisik sebesar 128MB. Sistem operasi yang ter-*install* pada komputer ini adalah *Windows* 2000 Professional Edition.

Komputer pertama yang digunakan sebagai *client* adalah komputer Bandulan dengan IP Address 10.126.17.254. Spesifikasi hardware yang digunakan pada komputer ini adalah AMD Duron dengan memori fisik sebesar 512 MB. Sistem operasi yang ter-*install* pada komputer ini adalah *Windows* 2000 Advanced Server.

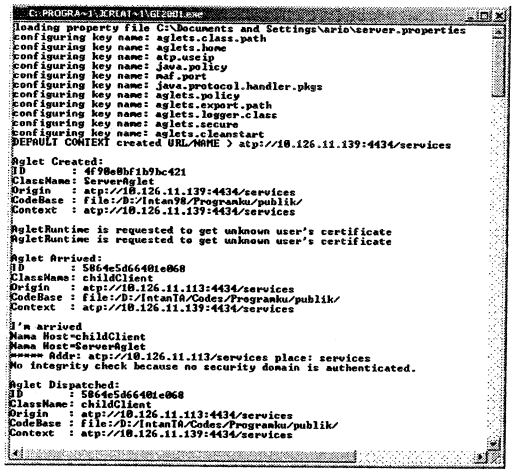
Komputer kedua yang digunakan sebagai *client* adalah komputer LP-13 dengan IP Address 10.126.11.113. Spesifikasi hardware yang digunakan pada komputer ini adalah Pentium III Celeron dengan memori fisik sebesar 128 MB. Sistem operasi yang ter-*install* pada komputer ini adalah *Windows* 2000 Professional Edition.

Uji Coba

Pada aplikasi ini diuji semua proses pada sistem perangkat lunak *agent.* Di antaranya adalah: *Peluncuran browser Internet Explorer, Penampilan Data halaman Web,* dan *Pengisian Polling halaman Web.*

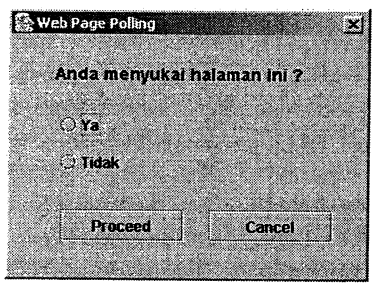
Pada pelaksanaan uji coba ini, aplikasi yang terlibat terdiri dari 2 sistem *agent.* Sistem *agent* yang pertama adalah aplikasi *server* yang dijalankan pada komputer LP-39. Sedangkan sistem *agent* yang kedua adalah aplikasi *client* yang dijalankan pada komputer LP-13 dan

Bandulan. Contoh hasil ujicoba ketika aglet client tiba :



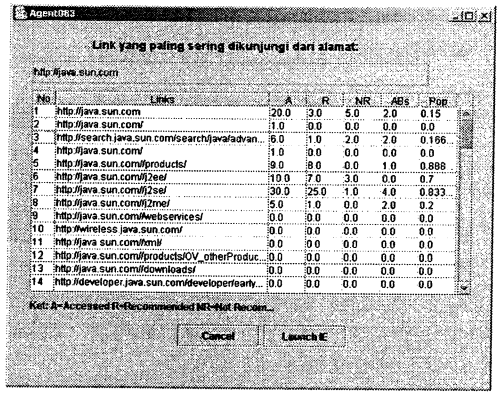
Gambar 8. Console Server Ketika Client Tiba

- Pengguna tidak memberikan polling ketika akan meninggalkan halaman Web yang sedang dikunjunginya
  Pengguna meninggalkan halaman Web yang sedang dikunjungi dengan mengklik salah satu *link* yang ada atau dapat langsung mengetikkan URL baru pada *address bar*. Sebelum *browser* me-*load* halaman Web yang baru, sebuah kotak dialog seperti gambar berikut akan muncul :



Gambar 9. Kotak Dialog Rekomendasi

- Pengguna melakukan selancar ke suatu halaman Web dan mendapatkan data rekomendasi atas halaman tersebut jika halaman tersebut pernah dikunjungi sebelumnya. Pengguna mengetikkan URL (misal java.sun.com) yang telah dikunjungi maka GUI client akan menampilkan data kunjungan alamat beserta seluruh *link* yang ada di dalamnya, seperti gambar berikut ini



No	Links	A	R	NR	ABs	Pop
1	http://java.sun.com	20.0	3.0	5.0	2.0	0.15
2	http://java.sun.com/	1.0	0.0	0.0	0.0	0.0
3	http://search.java.sun.com/search/java/advan...	6.0	1.0	2.0	2.0	0.166
4	http://java.sun.com/	1.0	0.0	0.0	0.0	0.0
5	http://java.sun.com/products/	9.0	8.0	0.0	1.0	0.888
6	http://java.sun.com/j2ee/	10.0	7.0	3.0	0.0	0.7
7	http://java.sun.com/j2se/	30.0	25.0	1.0	4.0	0.833
8	http://java.sun.com/j2me/	5.0	1.0	0.0	2.0	0.2
9	http://java.sun.com/#webservices/	0.0	0.0	0.0	0.0	0.0
10	http://wireless.java.sun.com/	0.0	0.0	0.0	0.0	0.0
11	http://java.sun.com/#xml/	0.0	0.0	0.0	0.0	0.0
12	http://java.sun.com/products/OV_otherProduc...	0.0	0.0	0.0	0.0	0.0
13	http://java.sun.com/#downloads/	0.0	0.0	0.0	0.0	0.0
14	http://developer.java.sun.com/developer/early...	0.0	0.0	0.0	0.0	0.0

Ket: A=Accessed R=Recommended NR=Not Recom...

Gambar 10. Hasil Rekomendasi Agent

Hasil dan Analisa Uji Coba

Dari uji coba yang telah dilakukan, terlihat bahwa perangkat lunak baik yang berupa aplikasi *client* maupun aplikasi *server* telah berjalan dengan baik. Komunikasi antara kedua sistem yang meliputi pengiriman data dan penerimaan data telah berjalan dengan baik.

Keberhasilan skenario-skenario tersebut ditunjukkan dengan keberhasilan komunikasi yang dilakukan antara aplikasi *client* sebagai *requester* rekomendasi halaman Web dengan aplikasi *server* sebagai penyedia database halaman Web. Komunikasi disini meliputi diterimanya aglet *client* pada sisi *server* serta *request* data halaman Web oleh aglet *client* kepada aglet *server*.

SIMPULAN

1. Dengan melakukan automasi Internet Explorer (IE) menggunakan JACOB, maka kontrol *WebBrowser* milik IE dapat ditangani dari dalam aplikasi Java. Ini juga berarti bahwa *browser* IE yang digunakan untuk melakukan selancar Web dapat diluncurkan dari dalam aplikasi Java.
2. Dengan menggunakan JACOB juga, semua *event* yang dimiliki *WebBrowser* dapat ditangkap agar dapat dilakukan suatu proses ketika ditemui *event* pengguna akan meninggalkan halaman yang sedang aktif. Dengan dasar ini, dapat dibangun sebuah modul untuk mencatat pendapat pengguna atas sebuah halaman Web.
3. Data pendapat pengguna atas sebuah halaman Web disimpan dalam sebuah database dan pengguna lain yang berselancar di Internet akan secara otomatis terhubung dengan aplikasi yang mengatur ekstraksi data dari database tersebut.

Dengan demikian, pengguna yang mengakses alamat atau URL yang terdapat dalam database akan mendapatkan rekomendasi berupa hasil statistik kunjungan dari alamat tersebut.

4. Dengan kemampuan komunikasi antar-aglet, maka aglet *client* dapat berkomunikasi dengan aglet *server*. Dengan demikian, aglet dapat digunakan sebagai pembawa pesan data alamat yang sedang dikunjungi pengguna dan juga pembawa rekomendasi hasil kunjungan dari alamat tersebut.

DAFTAR RUJUKAN

[1] Adler, D., *"The JACOB Project: A JAva-COm Bridge"*. September 2001 http://www.danadler.com/jacob

[2] Bigus, Joseph P. and Bigus, Jennifer. *"Constructing Intelligent Agents Using Java Second Edition"*. John Wiley and Sons, Inc. USA. 2001

[3] Karjoth, G., Ono, K., and Oshima, M. *"Aglets Specifications 1.1. Draft"*. IBM Tokyo Reasearch Laboratory, Japan, September 1998 http://www.trl.ibm.com/aglets/spec11.htm

[4] Lange B., Danny and Oshima, Mitsuru. *"Programming and Deploying Java™ Mobile Agents with Aglets™"*. Addison-Wesley Longman, Inc., Massachusets, USA, 1998.

[5] Liu, Jiming. *"Autonomous Agents and Multiagent Systems"*. World Scientific, London. 2001

[6] Microsoft. *"MSDN Library"*, Microsoft Corp. 2003 http://msdn.microsoft.com/library/default.asp

[7] Seanet Corporation. *"Internet for Beginners-Web Browsing"*. Seanet Corporation, Seattle, USA. http://www.seanet.com/help/general/web.shtml

[8] Skarmeas, Nikolaos. *"Agents as Objects with Knowledge Base State"*. Imperial College Press, London. 1999

[9] Sommers, B., *"Agents: Not just for Bond Anymore"*, JavaWorld Article, April 1997 http://www.javaworld.com/javaworld/jw-04-1997/jw-04-*agents*.html

[10] Venners, B., *"Under the Hood: The Architecture of Aglets"*, JavaWorld Article, April 1997 http://www.javaworld.com/javaworld/jw-04-1997/jw-04-hood.html

[11] Woolridge, Michael. *"An Introduction to MultiAgent Systems"*. John Wiley and Sons, Ltd., England. 2002