WEBDOWNLOADER

Agustinus Aruna Alangghya
Program Studi Sistem Informasi
Universitas Pelita Harapan Surabaya
carolusian@yahoo.com

Andreas Jodhinata
Program Studi Sistem Informasi
Universitas Pelita Harapan Surabaya
andreas.jodhinata@uphsurabaya.ac.id

***Abstrak* - Orang awam atau programmer pemula cukup kesulitan jika ingin mempelajari sebuah susunan *website* dari sebuah alamat tertentu dalam internet. Sehingga seringkali mereka harus menempuh proses manual untuk melakukan *copy* terhadap sebuah halaman HTML yang ingin dipelajari lebih lanjut. Skrispi ini disusun untuk memberikan kemudahan bagi orang awam atau programmer pemula sebab proses yang tadinya dilakukan secara manual dapat diotomasi oleh aplikasi *webdownloader* ini. Tidak terdapat sebuah algoritma khusus untuk mencapai tujuan skripsi ini. Sehingga algoritma yang disusun merupakan algoritma yang murni disusun sendiri agar keberhasilan aplikasi dapat tercapai. Menggunakan bahasa pemograman Java dan fasilitas Class-Class didalamnya sehingga proses otomasi yang dicari untuk *webdownloader* dapat tercapai. Aplikasi membutuhkan sambungan internet langsung untuk dapat melakukan proses *download* namun seluruh halaman yang berhasil di*download* dapat dinikmati secara *localhost* karena telah menggunakan *relative path*. Dari Hasil implementasi diketahui bahwa aplikasi *webdownloader* dapat diklaim berhasil menjalankan tujuan awal skripsi ini. Namun, memang tidak dapat melakukan download terhadap halaman *ajax* dan *embed object*.**

Kata Kunci - *Webdownloader*, *relative path*, *fixed path*, *stream*, koneksi, *webserver*.

I. PENDAHULUAN

Latar Belakang

Langkah pertama dalam belajar adalah meniru apa yang sudah ada sebelumnya. Hal yang sama pula harus dilakukan saat mengembangkan sebuah aplikasi berbasi *Web*. Meniru dari yang sudah baik terlebih dahulu, proses pembelajaran ini memakan waktu yang lama. Dapat disebut memakan waktu lama sebab pembelajaran tersebut dilakukan dalam 2 tahap.

Tahap pertama, melakukan *download* terhadap source code HTML, Javascript, CSS, dan konten lainnya untuk melakukan tahap kedua, yaitu, menelaah hasil *download* tersebut. Jika *download* dilakukan pada aplikasi berbasis *Web* yang memiliki jumlah *page* yang sedikit (misal kurang dari 10 *page*) maka proses manual untuk melakukan penyimpanan *page* secara manual dapat dilakukan. Namun, jika *Web* tersebut memiliki puluhan *page* maka tidak lumrah jika melakukannya dengan cara manual. Proses manual harus diganti dengan proses yang lebih terotomasi agar menghilangkan faktor waktu yang lama dan metode yang rumit dari proses manual tersebut.

Rumusan Masalah

Permasalahan yang diangkat dalam skripsi ini adalah :

1. Bagaimana membuat sebuah aplikasi yang mampu melakukan *download* secara otomatis seluruh halaman HTML dari sebuah alamat *Website* di internet?
2. Bagaimana mengubah *fixed path* dari situs *Website* menjadi *relative path* dalam komputer user?

Batasan Masalah

Aplikasi yang akan disusun memliliki batasan sebagai berikut :

1. Jika page memiliki tautan dengan *Web page*site lain, maka aplikasi akan berhenti melakukan pencarian dengan batasan : hanya mampu melewati 1 page di 1 buah *Web page*site loncatan saja.
2. Tidak melakukan *download* untuk *Webserver* yang melakukan pengamanan khusus, baik https ataupun setting *server* khusus yang tidak memungkinkan *Web downloade*r untuk men*download*nya
3. Tidak melakukan upload (layaknya FTP) untuk memperbaiki *Web* yang telah di*download*.
4. Tidak melakukan request secara *asynchronous* atau tidak dapat melakukan pengendalian terhadap halaman yang menggunakan AJAX.
5. Hanya akan membahas mengenai pemograman dari sisi klien saja, sebab pemecahan dari permasalahan tersedia mengambil dari pemograman dari sisi klien.

Tujuan Penelitian

Tujuan penelitian ini yakni membantu para developer pemula untuk bisa mempelajari susunan sebuah *Web* yang mereka anggap baik untuk dipelajari lebih lanjut.

Sistematika Penulisan

Laporan skripsi ini disusun dalam 6(enam) bab dengan perincian sebagai berikut :

BAB I PENDAHULUAN

Memberikan penjelasan awal mengenai aplikasi *Webdownload*er, rumusan masalah, tujuan pembuatan serta sistematika penulisan skripsi ini.

BAB II PEMOGRAMAN *WEB*

Pembahasan mengenai seluruh komponen yang menjadikan sebuah situs *Web* dapat bekerja normal. Serta memberikan penjelasan mengenai dynamic page dan *static page* serta implikasi mereka untuk tugas akhir ini.

BAB III PEMOGRAMAN *SOCKET*

Memberikan pembahasan menganai apa itu pemograman *socket* serta implementasinya dalam bahasa pemograman Java. Serta penjelasan mengenai dua protokol utama dalam pemograman *socket* .

BAB IV ALGORITMA *WEBDOWNLOAD*ER

Penyusunan algoritma untuk menyusun aplikasi Tugas Akhir ini, algoritma yang dihasilkan tidak meniru algoritma yang sudah ada sebelumnya untuk digunakan langsung namun, didesain khusus untuk mampu memecahkan permasalahan yang ada.

BAB V IMPLEMENTASI *WEBDOWNLOAD*ER

Pengunaan Bahasa Java untuk memberikan aplikasi yang dapat digunakan pada banyak platform. Serta memberikan pemecahaan masalah riil mengenai kebutuhan untuk memberikan aplikasi yang mampu melakukan *download* terhadap seluruh *Web page* yang dituju.

BAB VI PENUTUP

Pemberian kesimpulan dari aplikasi *Webdownload*er apakah mampu menjawab rumusan masalah yang diberikan pada BAB I. Hasil daripada penulisan skripsi ini terdapat pada bab ini.

II. PEMOGRAMAN WEB

WWW

World Wide Web (WWW) merupakan sebuah jaringan sumber informasi. Situs *Web* bersandar pada 3 konstruksi fundamental yang sudah tersedia agar situs tersebut dinikmati oleh lebih banyak pengguna.

Skema penamaan yang sama untuk mencari lokasi dari situs tersebut (contoh : URIs)

Protokol, agar memiliki kesamaan yang sama untuk mengakses sumber dari situs tersebut.

Hypertext, agar lebih mudah mengakses seluruh sumberdaya dalam situs[1].

Seluruh halaman dalam situs *Web* - dokumen HTML, gambar, video, dan lainnya – harus dialamatkan dan diencondingkan dengan *Universal Resource Identifier* atau URI.

URI terdiri dari 3 bagian :

1. Skema nama dari mekanisme yang digunakan sumber situs tersebut
2. Nama dari mekanisme hosting situs tujuan
3. Nama dari situs tujuan itu sendiri, disebut sebagai *path.*

URI dapat memberikan identitas pemisah dalam URI yang diberikan, biasanya diberikan dengan menambah tanda #. sebagai contoh, ini adalah URI penunjuk untuk sambungan bernama section_2, http://example.com#section_2[2]

Satu hal lain yang patut disimak adalah relative URI. Relative URI adalah pengalamatan secara umum kepada sebuah sumber yang masih ada dalam satu mesin yang sama.

Dalam WWW terdapat 2 jenis pemograman yakni :

1. Sisi Klien (*Client Side Scripting*)

Pemograman *Web* yang seluruhnya menggunakan sumber daya dari klien sehingga setiap klien bisa menikmati aplikasi tanpa memberikan beban kepada *server*.

Contoh : HTML, Javascript, CSS

2. Sisi *Server* (*Server Side Scripting*)

Pemograman *Web* yang menggunakan sumber daya dari *server* sehingga klien tidak melakukan proses lain selain menampilkannya ke user.

Contoh : PHP, JSP, Pearl, Ruby, Database Engine

HTML

HTML merupakan singkatan dari *HypreText Markup Language*, HTML merupakan standard penulisan dalam menyusun sebuah Web pagesite. Tag dalam HTML merupakan standard untuk menuliskan suatu tujuan tertentu.

HTML terdiri dari dari 3 bagian :

1. Baris yang memberi informasi tentang Versi HTML
2. Deklarasi bagian Header
3. Bagian Body yang mengandung isi dari HTML tersebut

[1] **W3C,** 1999, *HTML 4.01 Specification,* halaman 21

[2] **W3C,** 1999, *HTML 4.01 Specification,* halaman 22

contoh dari HTML yakni :

```
<!DOCTYPE>
<HTML>
   <HEAD>
      <TITLE>My       first       HTML
document</TITLE>
   </HEAD>
   <BODY>
      <P>Hello world! </BODY></HTML>
```

Untuk dapat melakukan fungsinya dengan baik dan benar maka HTML memiliki elemen-elemen khas yang dapat menjadikan sebuah Website yang dinamis dan berorientasi kepada penggunanya. Unik sebab, elemen harus ditulis secara sepasang, misal : <head></head> dan <a> selain ada beberapa elemen yang memang ditulis tunggal tidak berpasangan.

CSS

CSS merupakan singkatan untuk *Cascading Style Sheets*. CSS merupakan block penulisan untuk memberikan desain tampilan terhadap HTML. CSS secara khusus memilih tag-tag HTML tertentu untuk diberikan layout. Perubahan warna atau penebalan huruf dapat menjadi sebuah desain yang bagus untuk tampilan Web page secara keseluruhan.

Perkembangan CSS juga bisa menjadi perhatian tersendiri. Saat ini CSS telah mencapai versi 3.0. Perubahan radikal terjadi didalamnya, seperti :

1. animasi tanpa javascript
2. membuat layout koran
3. menjadikan border melengkung
4. membuat multiple background.

JAVASCRIPT[3]

Javascript merupakan bahasa scripting menggunakan akar bahasa Java namun semi-structured. Javascript atau kerap disebut js, memampukan Web browser untuk melakukan interaksi dengan user. Sebab js mampu menangkap input mouse dan keyboard lebih intens daripada <form> HTML dan CSS. Javascript mampu melakukan animasi seperti yang dilakukan oleh CSS hanya saja saat ini penggunaan animasi jika bisa dialihkan ke CSS maka CSS yang akan melakukannya. Hal ini dilakukan agar Web berjalan efisien.Dua hal yang layak disoroti untuk Javascript adalah

1. Penggunaan dalam tag <canvas> di HTML 5

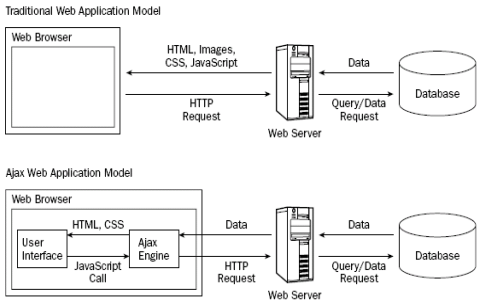
Pada HTML 5 terdapat canvas, yang memungkinkan melakukan penggambaran selayaknya Frame dalam Bahasa Pemograman Java. Javascipt membantu dalam pemilihan DOM, Document Object Model. Selain pemilihan DOM, js juga membantu melakukan penggambaran pada canvas tersebut dengan menggunakan method yang ada dalan js.

Contoh :

```
var              context              =
document.getElementByID('draw1')[0].g
etContext('2d');
context.strokeStyle='green';
context.strokeRect(10,10,50,50);
context.fillStyle='purple';
context.fillRect(70,70,50,50);
```

2. Penggunaan AJAX

AJAX merupakan singkatan dari Asynchronous Javascript and XML. Ajax memungkinkan sebuah tag html tertentu melakukan request ke Web server dan hanya pada bagian tag tertentu di refresh. Sehingga tidak seluruh page yang direfresh. AJAX memungkinkan sebuah Website lebih dinamis dan memakan bandwith yang lebih hemat.



Gambar 1: Diagram Cara Bekerja Ajax (http://www.techbubbles.com/aspnet/ajax-history/)

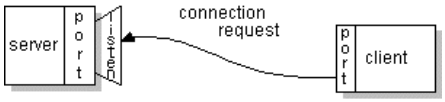
III. PEMOGRAMAN SOCKET

Socket adalah sebuah titik komunikasi yang memungkinkan terjadinya komunikasi antara minimal 2 program dalam satu *network* yang sama. Class *Socket* memberikan representasi koneksi antara *client* dan *server*. *Package* bawaan dari Java (java.net) memberikan 2 Class yakni *Socket* dan *ServerSocket* yang masing-masing mengimplementasikan koneksi dari *client* dan koneksi dari *Server*.

Secara normal, sebuah *server* dijalankan oleh sebuah komputer tersendiri dan memiliki *socket* yang terikat pada nomor *port* tertentu. *Server* hanya menunggu, mendengarkan dari *socket* untuk mengetahui bahwa ada klien yang sedang mencoba untuk melakukan koneksi ke *server*[4].

[3] JS, CSS, dan HTML merupakan intisari dari http://w3school.com

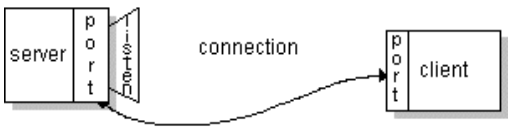
[4] http://docs.oracle.com/javase/tutorial/networking/sockets/definition.html diakses tanggal 20 Januari 2012, jam 13:10 WIB



Gambar 3.1 : Klien mencoba melakukan koneksi ke server (http://docs.oracle.com/javase/tutorial/figures/networking/5connect.gif)

Klien akan mencari dan mendapatkan *hostname* dari *server* dan nomor *port* berapa *server* tersebut yang terbuka. Untuk dapat melakukan permintaan koneksi, klien akan berusaha bertemu dengan nomor *port server* tersebut dan mengindentifikasi dirinya sehingga keduanya dapat berkomunikasi dari 1 *port* yang sama. Seperti digambarkan pada gambar 3.

Kemudia klien dapat berkomunikasi dengan *server* dengan cara menulis atau membaca dari *socket* yang telah tersambung. Pada gambar 4, Ketika koneksi telah terjadi, maka *server* segera akan membuat *socket* baru yang nantinya akan melakukan remote terhadap alamat soket asal, inilah yang menjadikan *server* dapat mengasosiasikan diri dengan klien dan nantinya tetap dapat menerima koneksi yang baru dari klien lain.



Gambar3.2: Koneksi terjadi antara server dan klien (http://docs.oracle.com/javase/tutorial/figures/networking/6connect.gif)

Pemograman *Socket* tersebut digunakana apabila membuat sebuah aplikasi yang membutuhkan *Socket* khusus untuk dapat digunakan baik dan benar. Java memberikan satu cara lain yang dapat mengakses sebuah alamat *Website* tanpa harus mengetahui *socket* yang tersedia dalam *Webserver* bahkan harus membuat aplikasi *server* terlebih dahulu.

IV. ALGORITMA WEBDOWNLOADER

*Webdownload*er membutuhkan dua metode untuk dapat dijalankan dengan sempurna. Metode tersebut yaitu :

a. Metode pemotongan baris HTML
b. Metode mengubah *fixed path* menjadi *relative path*

dengan dua metode tersebut, dipastikan *Webdownload*er dapat berjalan dan berfungsi dengan sempurna.

Metode Pemotongan Baris HTML

Input yang diterima setelah melakukan koneksi ke *Web server* merupakan kumpulan teks yang merupakan susunan HTML murni dari *Web* tersebut. Untuk dapat mendeteksi seluruh tautan yang ada pada *Web page* tersebut, maka harus dilakukan penyaringan. Penyaringan difokuskan pada *attribute* href dan src pada seluruh isi HTML. Sebab pada kedua *attribute* inilah terdapat tautan menuju ke halaman, gambar atau video pada *Web page* selanjutnya. Aplikasi haruslah mampu melakukan pemotongan pada tautan yang diberikan diantara dua tanda-petik pada dua *attribute* tersebut. Sebagai contoh :

```
<a href="http://www.w3schools.com">
<img src="smiley.gif" alt="Smiley face"/>
```

Pada dua bagian diatas harus menghasilkan, http://www.w3schools.com dan http://www.w3schools.com/smiley.gif

Metode mengubah *fixed path* menjadi *relative path*

Fixed Path adalah tautan baku yang didapatkan setiap kali melakukan koneksi ke *Web server*, tautan baku ini merupakan alamat sebenarnya dari *Web page* yang dituju. Masalah yang muncul jika hanya menggunakan *fixed path* adalah tingkat ketergantungan yang tinggi terhadap *Web-server* atau dengan kata lain, hanya dapat dibuka jika tersedia jaringan internet untuk mengakses *Web server*. Padahal, tujuan dari penggunaan *Webdownload*er ini adalah untuk memampukan user melakukan *browsing offline* tanpa perlu ada jaringan internet.

Solusi untuk ketergantungan terhadap *server* adalah *relative path*. *Relative path* adalah jalur pengalamatan terhadap direktori tertentu. Sehingga *Web page* dapat dilihat secara *offline*. Metode ini dituangkan dalam fungsi Penyimpanan*Web*_SecaraLokal.

Susunan Algoritma

Algoritma disusun secara umum agar layak digunakan dalam bahasa pemograman apapun, sehingga bersifat umum dan tidak mengacu pada satu bahasa pemograman saja.

Fungsi koneksi_*Webserver*

Fungsi ini digunakan untuk mendapatkan setiap baris HTML standard dari URL yang diinputkan. Bertujuan untuk dapat menampilkan HTML secara utuh dan lengkap dari URL yang telah diinputkan.

Algoritma dari fungsi koneksi_Webserver adalah sebagai berikut :

1. Input URL yang dituju.
2. Melakukan koneksi ke *Web server* menggunakan *port* 80 sesuai input URL
3. Jika koneksi berhasil dilakukan maka lakukan langkah 4. Jika tidak maka lakukan langkah 6.
4. Membuka jalur *stream* untuk dapat menangkap *output* yang diberikan oleh *Webserver*.
5. Menangkap setiap *stream* yang dihasilkan untuk disimpan kedalam String input.
6. Menutup *Stream* dan memutuskan koneksi ke *Webserver*

Fungsi PencarianURL

Fungsi ini digunakan saat melakukan koneksi dan melakukan pencarian pada setiap baris HTML untuk mencari semua tautan didalamnya.

Algoritma dari fungsi PencarianURL adalah sebagai berikut :

1. Melakukan **Fungsi koneksi_*Webserver***.
2. Mengecek apakah dalam String input terdapat pola kata ‘href=’ atau ‘src=’
3. Jika terdapat pola kata ‘href=’ atau ‘src=’ maka String akan dipotong dengan ketentuan :
   - StrLanjutan=substring(*stream*,indexAwalPola+5,indexTandaPetikTerdekat) untuk src
   - StrLanjutan=substring(*stream*,indexAwalPola+6,indexTandaPetikTerdekat) untuk href.
4. Menyimpan String keluaran langkah 6 kedalam ArrBranch
5. Kembali ke langkah 5 hingga tidak ada lagi pola kata ‘href=’ atau ‘src=’
6. Ulangi langkah 2-8 hanya kali ini dengan setiap isi dari ArrBranch, perulangan berhenti ketika sudah tidak ada lagi tambahan yang masuk ke dalam ArrBranch

Setelah selesai mendapatkan seluruh tautan dalam URL yang diinputkan maka tahap selanjutnya adalah mengubah pengalamatan yang ada dalam setiap halaman *Web* tersebut untuk dapat dibuka secara lokal. Pengubahan ini merupakan bagian yang penting agar terjadi konsistensi antara *Web* yang dibuka secara lokal maupun *Web* sebenarnya secara *online*.

Fungsi Penyimpanan*Web*_SecaraLokal

Algoritma dari fungsi Penyimpanan*Web*_ SecaraLokal adalah sebagai berikut :

1. Memasukan namaProject kedalam variable namaProject
2. Melakukan Langkah 2-9 selama isi ArrBranch belum habis.
3. _String inputLine ←isi ArrBranch
4. Jika inputLine mengandung pola “.jpeg”, “.jpg”, “.bmp”, “.gif”, dan “.png” maka langsung menuju Menyimpan dengan Algoritma_saveByte dan langsung menuju Langkah 10
5. Melakukan Fungsi koneksi_*Webserver*
6. Jika terdapat pola kata ‘href=’ atau ‘src=’ maka String akan dipotong dengan ketentuan :

```
String _normalizedLine = stream.replace("http://",
"/"+this.namaProject +"/");
```

7. Setelah tidak ada lagi baris *stream* maka selanjutnya adalah memberi nama file agar bisa disimpan sesuai dengan path yang tersedia.
8. Membuat direktori yang sesuai dengan susunan String
9. Jika pada String inputLine tidak mengandung ekstensi file apapun maka dilakukan hal berikut :

```
inputLine +=".html";
```

10. Namun jika String inputLine sudah mengandung ekstensi file maka langsung menyimpan dengan nama file sesuai dengan String inputLine.

Fungsi saveByte

Memungkinkan sebuah image atau object lainnya untuk disimpan dari Webserver dengan tetap mempertahankan keaslian object tersebut. Algoritma Fungsi saveByte adalah sebagai berikut :

1. Membuka *Stream* baru untuk menyimpan input yang diberikan dengan catatan tetap mempertahankan *stream* tersebut dalam bentuk byte dan tidak mengubahnya menjadi String
2. *Stream* tersebut dibuka dengan kapasitas maksimal 1024byte
3. Simpan input kedalam *stream* selama masih ada byte yang masih dapat dibaca atau disebut sebagai *buffered stream*.
4. Mengembalikan *stream* dalam bentuk array of byte

V. KESIMPULAN

Kesimpulan yang dapat diberikan setelah mengimplementasikan aplikasi *Webdownload*er berdasarkan permasalahan dan tujuan penelitian yang diberikan pada skripsi ini yaitu :

1. Secara garis besar tujuan penelitian ini dapat tercapai sebab aplikasi *webdownloader* sungguh membantu developer pemula dalam mempelajari struktur web yang dirasa baik untuk dipelajari lebih lanjut.
2. Tidak dapat melakukan pelacakan atau penyimpanan halaman *web* yang mengandung AJAX.
3. Tidak dapat melakukan penyimpanan terhadap <frame> atau *embed object* yang merupakan sebuah obyek rumit dan biasanya dilindungi oleh *web server* asal

REFERENSI

[1] Cahyadi,Daniel. Kiswono Prayogo. Modul Praktikum Java. 2011. Surabaya: UPH Surabaya.

[2] Geogia State University. Pembahasan Java . Tersedia di *http://www2.gsu.edu/~matknk/java/reg97-5.htm*; Internet; diakses pada tanggal 21 Januari 2012 pukul 12:35 WIB

[3] Java Community. Catalog Java. Tersedia di *http://java2s.com/Tutorial/Java/CatalogJava.htm*; Internet; diakses pada tanggal 10 Januari 2012 pukul 07:55 WIB

[4] Oracle Webserver. Class URL. Tersedia di *http://docs.oracle.com/javase/1.4.2/docs/api/java/net/URL.html*; Internet; diakses pada tanggal 7 Januari 2012 pukul 10:11 WIB

[5] Ragged, Dave. Arnaud Le Hors. Ian Jacobs. HTML 40. 1999. Europe: W3C

[6] World Wibe Web Consortium. Online-web-tutorial. Tersedia di *http://www.w3schools.com/*; Internet; diakses pada tanggal 7 Januari 2012 pukul 10:45 WIB