SKRIPSI

WEBDOWNLOADER

Ditulis untuk memenuhi sebagian persyaratan akademik guna memperoleh
gelar Sarjana Komputer Strata Satu

Oleh :

NAMA : AGUSTINUS ARUNA ALANGGHYA

NPM : 08120080006

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PELITA HARAPAN
SURABAYA

2012

PERNYATAAN KEASLIAN KARYA TUGAS AKHIR

Saya mahasiswa Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Pelita Harapan Surabaya,

Nama Mahasiswa : Agustinus Aruna Alangghya
Nomor Pokok Mahasiswa : 08120080006
Program Studi : Sistem Informasi

Dengan ini menyatakan bahwa karya tugas akhir yang saya buat dengan judul "**WEBDOWNLOADER**"
adalah :

1) Dibuat dan diselesaikan sendiri, dengan menggunakan hasil kuliah, tinjauan lapangan dan buku–buku serta jurnal acuan yang tertera di dalam referensi pada karya tugas akhir saya.
2) Bukan merupakan duplikasi karya tulis yang sudah dipublikasikan atau yang pernah dipakai untuk mendapatkan gelar sarjana di universitas lain, kecuali pada bagian–bagian sumber informasi dicantumkan dengan cara referensi yang semestinya.
3) Bukan merupakan karya terjemahan dari kumpulan buku atau jurnal acuan yang tertera di dalam referensi pada karya tugas akhir saya.

Kalau terbukti saya tidak memenuhi apa yang telah dinyatakan di atas, maka karya tugas akhir ini batal.

Surabaya, 1 Mei 2012
Yang membuat pernyataan

Materai
Rp 6000

(A. ARUNA A.)

UNIVERSITAS PELITA HARAPAN SURABAYA

FAKULTAS ILMU KOMPUTER

PERSETUJUAN DOSEN PEMBIMBING TUGAS AKHIR

"WEBDOWNLOADER"

Oleh :

Nama	**: Agustinus Aruna Alangghya**
NPM	**: 08120080006**
Program Studi	**: Sistem Informasi**

Telah diperiksa dan disetujui untuk diajukan dan dipertahankan dalam Sidang Tugas Akhir guna memperoleh gelar Sarjana Komputer Strata Satu pada Program Studi Sistem Informasi, Fakultas Ilmu Komputer Universitas Pelita Harapan, Surabaya, Jawa Timur.

Surabaya, 3 Mei 2012

Menyetujui :

Pembimbing Skripsi

(Andreas Jodhinata, S.Kom, M.Kom)

Ketua Program Studi	**Dekan Fakultas**
Sistem Informasi	**Ilmu Komputer**
(Andreas Jodhinata, S.Kom, M.Kom)	**(Prof. Dr. Ir. Kuswara Setiawan, M.T)**

UNIVERSITAS PELITA HARAPAN SURABAYA
FAKULTAS ILMU KOMPUTER

PERSETUJUAN TIM PENGUJI TUGAS AKHIR

Pada hari Selasa, 22 Mei 2012 telah diselenggarakan Sidang Tugas Akhir untuk memenuhi sebagian persyaratan akademik guna memperoleh gelar Sarjana Komputer Strata Satu pada Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Pelita Harapan Surabaya, atas nama :

Nama	**: Agustinus Aruna Alangghya**
NPM	**: 08120080006**
Program Studi	**: Sistem Informasi**
Fakultas	**: Ilmu Komputer**

termasuk ujian Tugas Akhir yang berjudul "WEBDOWNLOADER" oleh tim penguji yang terdiri dari :

Nama Penguji	Jabatan dalam Tim Penguji	Tanda Tangan
1. Andreas Jodhinata, S.Kom, M.Kom. (Penguji I)	, sebagai Ketua	_____
2. Donald Latumahina, S.Kom, M.Comp (Penguji II)	, sebagai Anggota	_____
3. Prof.Dr.Ir.Kuswara Setiawan, M.T (Penguji III)	, sebagai Anggota	_____

Surabaya, 22 Mei 2012

ABSTRAK

Agustinus Aruna Alangghya (08120080006)

WEBDOWNLOADER
(xi + 31 halaman; 12 gambar; 5 tabel)

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*.

Referensi : 6 (2003 - 2012)
Kata Kunci : *Webdownloader*, *relative path*, *fixed path*, *stream*, koneksi, *webserver*.

ABSTRACT

Agustinus Aruna Alangghya (08120080006)

WEBDOWNLOADER
(xi + 31 pages; 12 pictures; 5 tables)

Beginner or Earlier programmer have a problem when studying HTML. They would try create a page not from scratch but from the web page which was appeared in Internet. In this occasion, they copied those web page manually and one by one. This thesis will automate those process. User just need to type a single domain name and click process button, afterward all webs would have been downloaded.

This thesis doesn't have any special algortyhm which is ever exist before. Algortyms are created on demand to fullfil this thesis purpose. Using Java to create the application, this Thesis's application using Java Classes to acquire the goal of the Thesis. All downloaded pages can be shown in localhost because changes of its native, from fixed path into relative path.

Implementation show that This thesis can claim the success. But application can not download the page which has AJAX page and embed object.

References : 6 (2003 - 2012)
Key Words : *Webdownloader*, *relative path*, *fixed path*, *stream*, koneksi, *webserver*.

KATA PENGANTAR

Puji dan syukur saya haturkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan anugerah-Nya sehingga tugas akhir ini dapat terselesaikan dengan baik dan tepat waktu.

Tugas Akhir dengan judul "WEBDOWNLOADER" ini ditujukan untuk memenuhi sebagian persyaratan akademik guna memperoleh gelar Sarjana Komputer Strata Satu Universitas Pelita Harapan, Surabaya.

Demikian pula atas segala bimbingan, dorongan, motivasi yang amat bermanfaat sejak awal kuliah sampai terbitnya tugas akhir ini dari :

1. Yang terhormat Prof.Dr.Ir.Kuswara Setiawan, M.T, selaku Dekan Fakultas Ilmu Komputer.
2. Yang terhormat Bapak Andreas Jodhinata, S.Kom, M.Kom, selaku Ketua Program Studi Sistem Informasi dan Dosen Pembimbing.
3. Yang terhormat Bapak David Sundoro, S.T, M.MT, selaku Penasehat Akademik.
4. Semua dosen yang telah memberikan ilmu dan bimbingan selama berkuliah di Universitas Pelita Harapan Surabaya.
5. Yang terkasih Ayah, Ibu, dan Adik-Adik serta seluruh keluarga besar.
6. Rekan-rekan se-Almamater di Universitas Pelita Harapan Surabaya terutama Program Studi Sistem Informasi.

Saya sampaikan terima kasih yang paling tulus.

Semoga tugas akhir ini dapat bermanfaat bagi orang lain dan saran perbaikan selalu dinantikan.

Surabaya, 1 Mei 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bagian ini berisi informasi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, dan sistematika penulisan.

1.1 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.

1.2 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?

1.3 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 download*er 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.

1.4 Tujuan Penelitian

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

1.5 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.

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.

2.1 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]

---

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

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

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

2.2 HTML

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

HTML terdiri dari dari 3 bagian[3] :

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

contoh dari HTML yakni :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
   <HEAD>
      <TITLE>My first HTML document</TITLE>
   </HEAD>
   <BODY>
      <P>Hello world! </BODY></HTML>
```

[3] http://www.w3.org/TR/html401/struct/global.html#h-7.1 diakses tanggal 11 Januari 2012 jam 12.05 WIB

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.

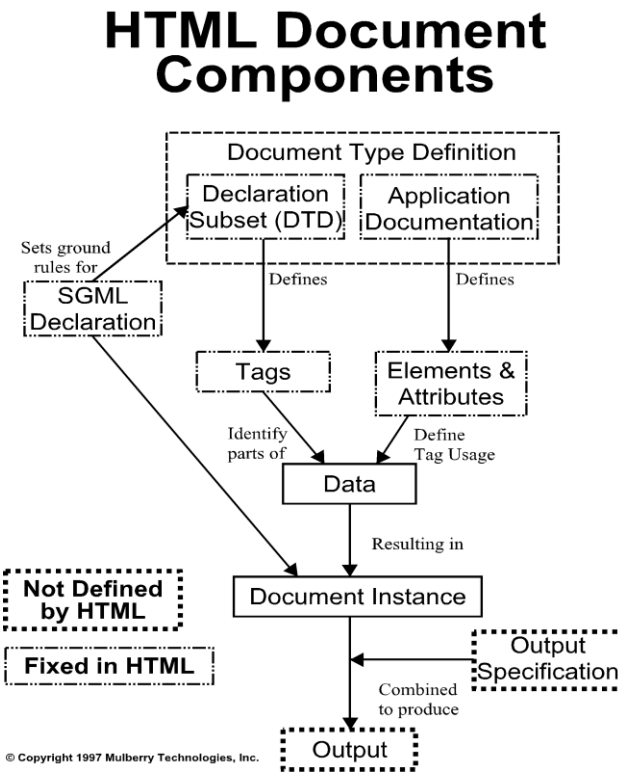
Elemen-elemen tersebut antara lain[4] :

Tabel 2.1 : Elemen-elemen HTML

Nama Tag	Tujuan
<html>	Membuat struktur html dasar
<head>	Memberikan informasi serta inisialisasi mengenai HTML tersebut.
<p>	Membuat paragraph
<img>	Memasukan gambar
<table>	Membuat tabel
<tr> <td>	Biasanya digunakana sepasang untuk membuat baris dan kolom
<form>	Membuat form agar dapat digunakan secara aktif, menggunkana metode GET atau POST
<input>	Memberikan fasilitas input kepada user
<a>	Memberikan tautan kepada halaman tersebut
	Memberikan pengurutan dan diberi penomoran
	Memberikan pengurutan tanpa penomoran

Dalam hubungan dengan Tugas akhir ini, maka yang menjadi perhatian utama adalah tag HTML sebab inilah kunci untuk dapat melakukan pencarian lanjutan dari *page* asal menuju ke *page* tujuan. Selain itu, terdapat juga selector *src* yang juga harus ditelusuri untuk mendapatkan seluruh bagian yang ditautkan pada *Web page* tersebut.

[4] Diintisarikan dari http://www.w3schools.com/html/default.asp diakses 14 Januari 2012 jam 14.30 WIB



Gambar 2.1: Diagram HTML Standard Beserta Seluruh Elemen Didalamnya

Gambar 2.1 merupakan topologi HTML yang dapat menggambarkan dengan jelas bagaimana HTML saat ini digunakan[5]. DTD memberikan definisi terhadap Tag yang akan dibangun, sedangkan kebutuhan aplikasi (*application documents*) memberikan definisi terhadap element dan attribute apa saja yang dipasangkan ke dalam tag. Keduanya menjadi sebuah data, dan datum dianggap menjadi sebuah obyek HTML utuh. HTML utuh inilah yang dinikmati oleh user.

2.3 CSS

CSS merupakan singkatan untuk *Cascading Style Sheets*. CSS merupakan block penulisan untuk memberikan desain tampilan terhadap HTML.

[5] http://www.students.tut.fi/~leppane7/leppanen.html diakses tanggal 14 Januari 2012 jam 15.15 WIB

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 :

- animasi tanpa javascript
- membuat layout koran
- menjadikan border melengkung
- membuat multiple background

contoh CSS :

```
.fbEmu .body .fbEmuLink{color:#333}
.fbEmu .body .fbEmuLink:hover{text-decoration:none}
.fbEmu .body a.signature{color:#3b5998;display:inline}
.fbEmu .body a.signature:hover{text-decoration:underline}
```
[6]

Hal yang patut disayangkan, belum ada keseragaman antar *Web* browser yang menjadikan setiap *Web* browser (secara garis besar, aliran Mozilla dan aliran Chrome) memiliki tag sendiri-sendiri dalam penulisan kekhususan CSS tersebut. Contoh Beberapa Tag berbeda tersebut yaitu :

- Firefox :
  + @-moz-keyframes [nama_function]{}
  + -moz-transform //memungkinkan untuk melakukan efek putar pada tag html
  + -moz-transition //memungkinkan transisi sebuah tag html
- Google Chrome :
  + *Web*-kit
  + *Web*kit-transition //memungkinkan transisi sebuah tag html

[6] Cuplikan CSS dari *Wall Facebook*, mirror : http://static.ak.fbcdn.net/rsrc.php/v1/yS/r/FGEDDUmn2cg.CSS

2.4 JAVASCRIPT

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

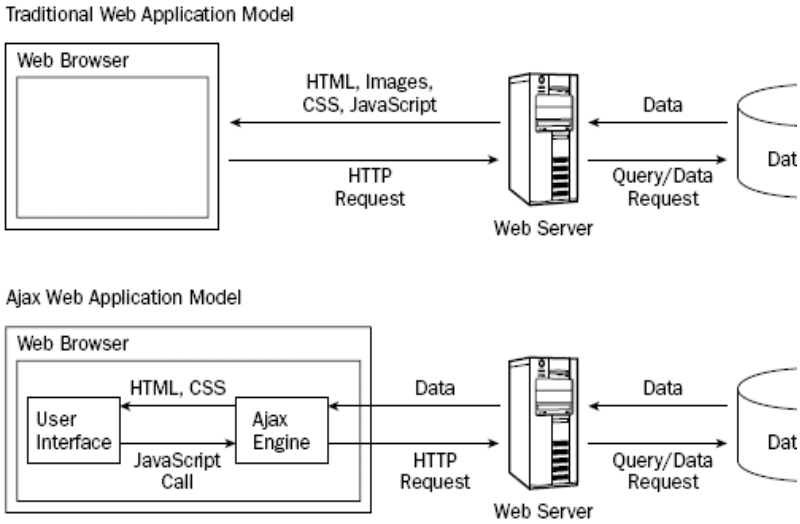
   Pada HTML 5 terdapat canvas, yang memungkinkan melakukan penggambaran selayaknya Frame dalam Bahasa Pemograman Java. Javascipt membantu dalam pemilihan DOM, *Document Object Model*. Selain pemilihan DOM, js juga membantu melakukan penggambaran pada canvas tersebut dengan menggunakan method yang ada dalan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

   Contoh :

```
var context =
document.getElementByID('draw1')[0].getContext('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 di*refresh*. AJAX memungkinkan sebuah *Website* lebih dinamis dan memakan *bandwith* yang lebih hemat.



Gambar 2.2: Diagram Cara Bekerja Ajax
(http://www.techbubbles.com/aspnet/ajax-history/)

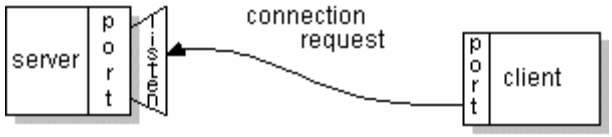
BAB III
PEMOGRAMAN *SOCKET*

Memberikan pembahasan menganai apa itu pemograman *socket* serta implementasinya dalam bahasa pemograman Java. Menjelaskan bagain yang penting untuk membangun aplikasi *Webdownloader*.

3.1 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*[7].

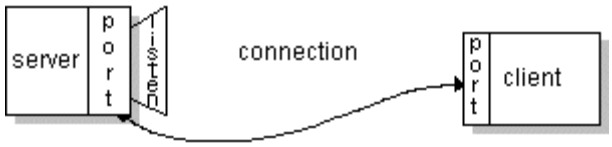


Gambar 3.1 : Klien mencoba melakukan koneksi ke server
(http://docs.oracle.com/javase/tutorial/figures/networking/5connect.gif)

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

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.

3.2 Class Java.Net.*Socket*

Memberikan penjelasan mengenai class *socket* dalam Java, namun dikarenakan bagian ini tidak digunakan dalam implementasi aplikasi *Web downloade*r, maka class *Socket* hanya akan disebutkan tanpa dijelaskan.

Tabel 3.1 Method Socket pada Java

public void	**bind**(*Socket* Address bindpoint) throws IOException
public void	**close**() throws IOException
public void	**connect**(*Socket* Address endpoint) throws IOException
public void	**connect**(*Socket* Address endpoint, int timeout) throws IOException
public void	**sendUrgentData**(int data) throws IOException
public void	**shutdownInput**() throws IOException
public void	**shutdown*Output***() throws IOException
public String	**toString**()

Tabel 3.2 Properties Socket pada Java

public boolean	**isBound**()
public *Socket* Channel	**getChannel**()
public boolean	**isClosed**()
public boolean	**isConnected**()
public InetAddress	**getInetAddress**()
public boolean	**isInputShutdown**()
public *OutputStream*	**get*OutputStream***() throws IOException
public void	**setPerformancePreferences**(int connectionTime, int latency, int bandwidth)
public int	**get*Port***()
public void	**setReceiveBufferSize**(int size) throws *Socket* Exception
public int	**getReceiveBufferSize**() throws *Socket* Exception
public *Socket* Address	**getRemote*Socket* Address**()
public void	**setReuseAddress**(boolean on) throws *Socket* Exception

public boolean	**getReuseAddress**() throws *Socket* Exception
public void	**setSendBufferSize**(int size) throws *Socket* Exception
public int	**getSendBufferSize**() throws *Socket* Exception
public void	**setSoLinger**(boolean on, int linger) throws *Socket* Exception
public int	**getSoLinger**() throws *Socket* Exception
public void	**setSoTimeout**(int timeout) throws *Socket* Exception
public int	**getSoTimeout**() throws *Socket* Exception
public void	**setTcpNoDelay**(boolean on) throws *Socket* Exception
public boolean	**getTcpNoDelay**() throws *Socket* Exception
public void	**setTrafficClass**(int tc) throws *Socket* Exception
public int	**getTrafficClass**() throws *Socket* Exception

3.3 Class Java.Net.URL

Bagian inilah yang digunakan untuk menyusun aplikasi *Web downloade*r sebab fungsinya yang memberikan koneksi dari *Webserver* serta mengambil data yang diperlukan.

Tabel 3.3 Method Class URL pada Java

public boolean	**equals**(Object obj) memberikan perbandingan apakah object yang dibandingkan juga objek URL
public int	**hashCode**() megembalikan URL dalam keadaan terenkripsi
public URLConnection	**openConnection**() throws IOException

	membuka koneksi baru ke *Web server*, memungkinkan aplikasi tersambung ke *Web server*
public URLConnection	**openConnection**(Proxy proxy) throws IOException membuka sambungan sama seperti diatas hanya saja menggunakan paramater Obyek dari Proxy
final public Input*Stream*	**open*Stream***() throws IOException mengambil *stream* yang ada pada *Web server*
public boolean	**sameFile**(URL other) membandingkan obyek dengan file dari URL yang diberikan apakah mereka adalah sama
protected void	**set**(String protocol, String host, int *port*, String file, String ref) melakukan setting spesial untuk koneksi dengan *Web server*
protected void	**set**(String protocol, String host, int *port*, String authority, String userInfo, String path, String query, String ref) melakukan setting spesial untuk koneksi dengan *Web server*

public String	**toExternalForm**() memberikan cetakan String kepada peminta
public String	**toString**()
public URI	**toURI**() throws URISyntaxException mengubah semua URL yang sesuai dengan RFC 2396 menjadi URI

Tabel 3.4 Properties Class URL pada Java

public String	**getAuthority**() mengembalikan perihal otoritas dari URL
public int	**getDefault*Port***() mengembalikan nomor *port Web server*
public String	**getFile**() memunculkan bagian file dari URL
public String	**getHost**() menyebutkan host dari *Web server*
public String	**GetPath**() memunculkan *path* dari URL
public int	**get*Port***() mengembalikan nomor terkahir *port Web server*
public String	**getProtocol**() menyebutkan protocol *Web server*
public String	**getQuery**() menyebutkan quersy String dalam URL
public String	**getRef**() mengambil *anchor* dalam URL
public static void	**setURL*Stream*HandlerFactory**(URL*S tream*HandlerFactory fac) memastikan bahwa JVM diijinkan untuk melakukan koneksi

public String	**getUserInfo**() mengambil user info dari URL
final public Object	**getContent**(Class[] classes) throws IOException mengambil content dari URL
final public Object	**getContent**() throws IOException mengambil content dari URL

BAB IV
ALGORITMA *WEBDOWNLOAD*ER

Bab ini memberikan penjelasan secara mendetail mengenai algoritma yang akan digunakan demi mewujudkan aplikasi *Webdownload*er yang berfungsi sempurna.

*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.

4.1 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`

4.2 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.

4.3 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 :

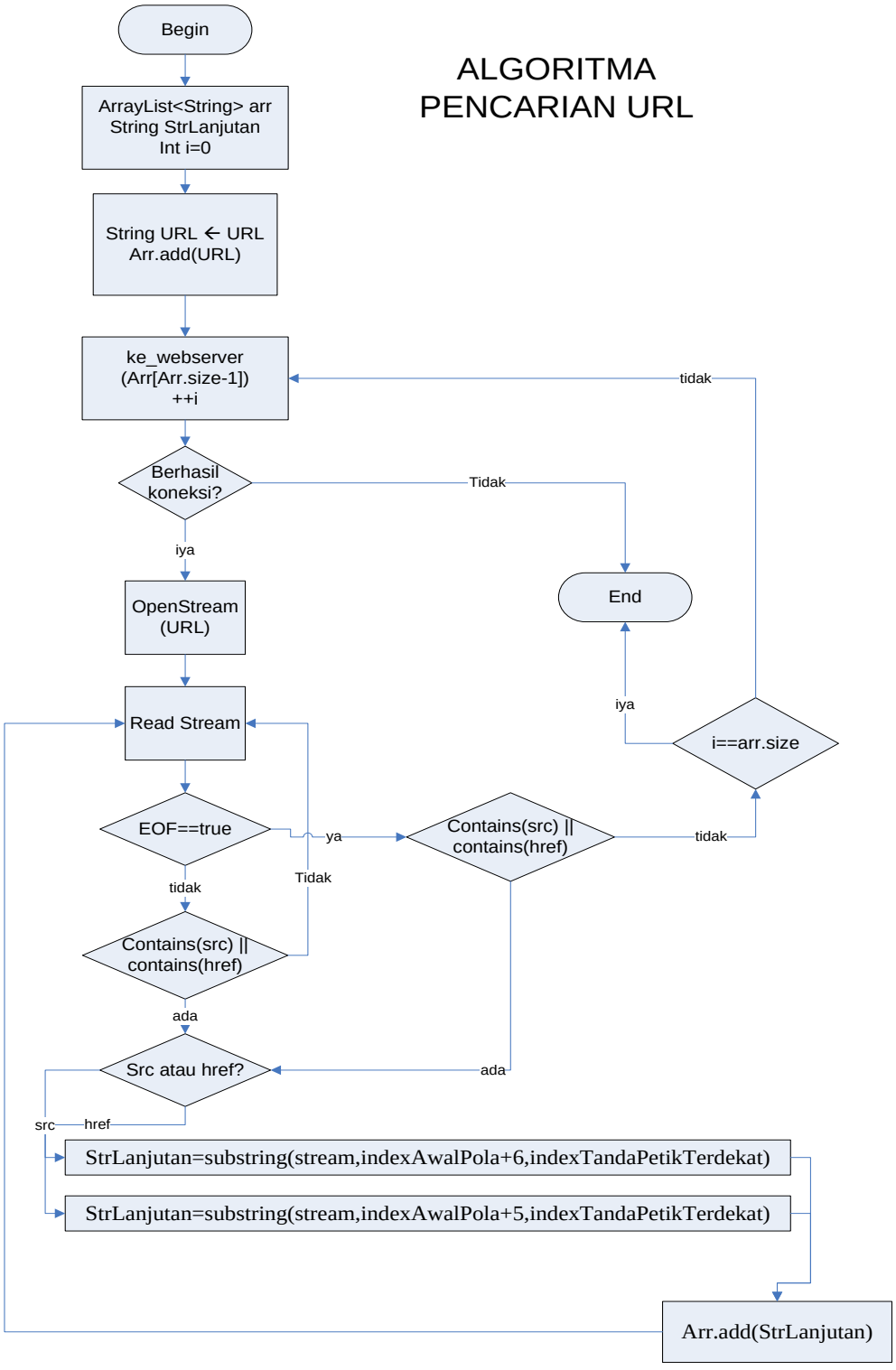
1. Membuka *Stream* baru untuk menyimpan input yang diberikan dengan catatan tetap mempertahankan *stream* tersebut dalam bentuk byte dan tidak mengubahnya menjadi String
2. *Stream* tersebut dibuka dengan kapasitas maksimal 1024byte

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

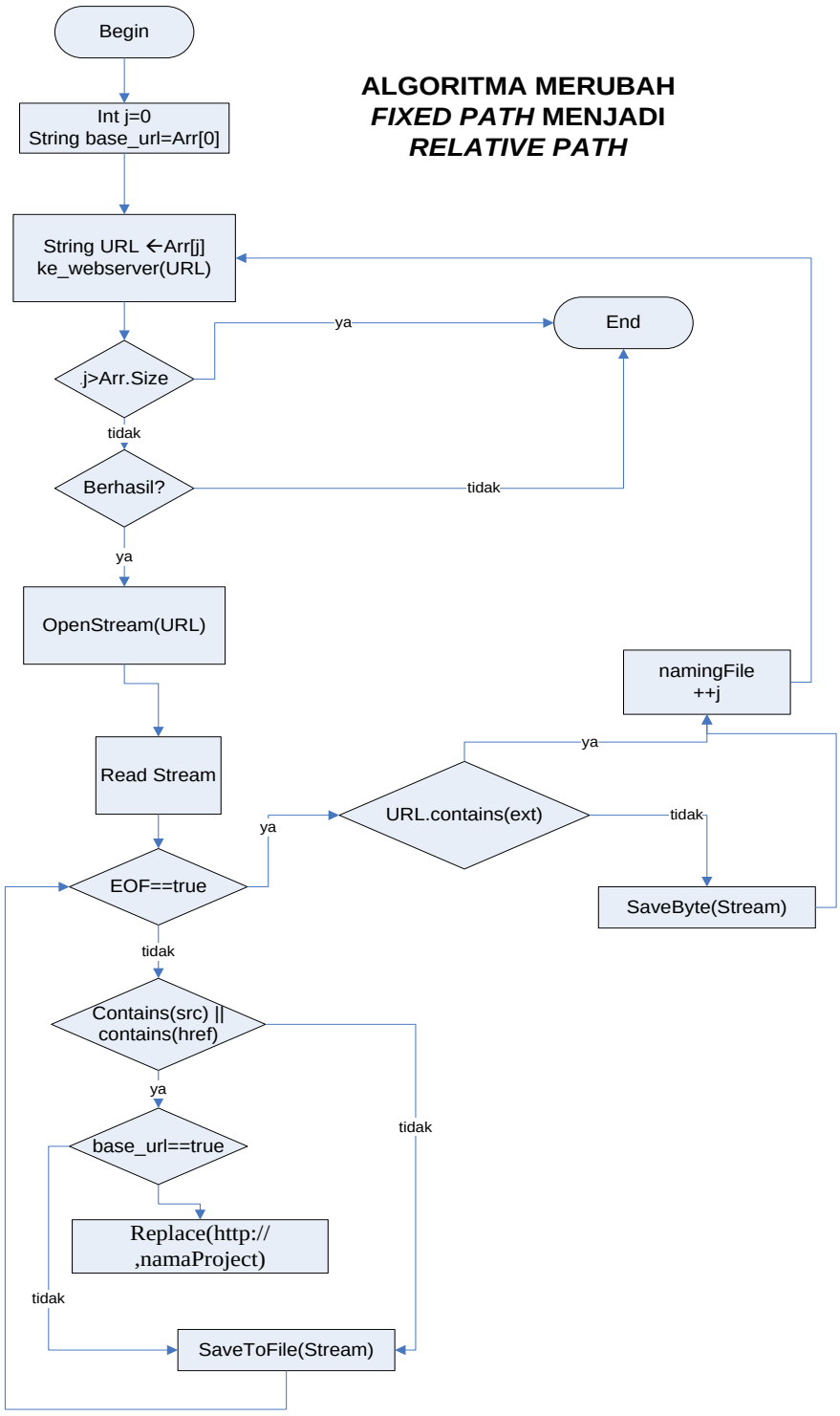
4.4 Bagan Algoritma

Algoritma Pencarian URL sebagaimana yang dijelaskan pada sub bab 4.3 dapat digambarkan secara detail menggunakan *flowchart* seperti yang tampak pada gambar 4.1

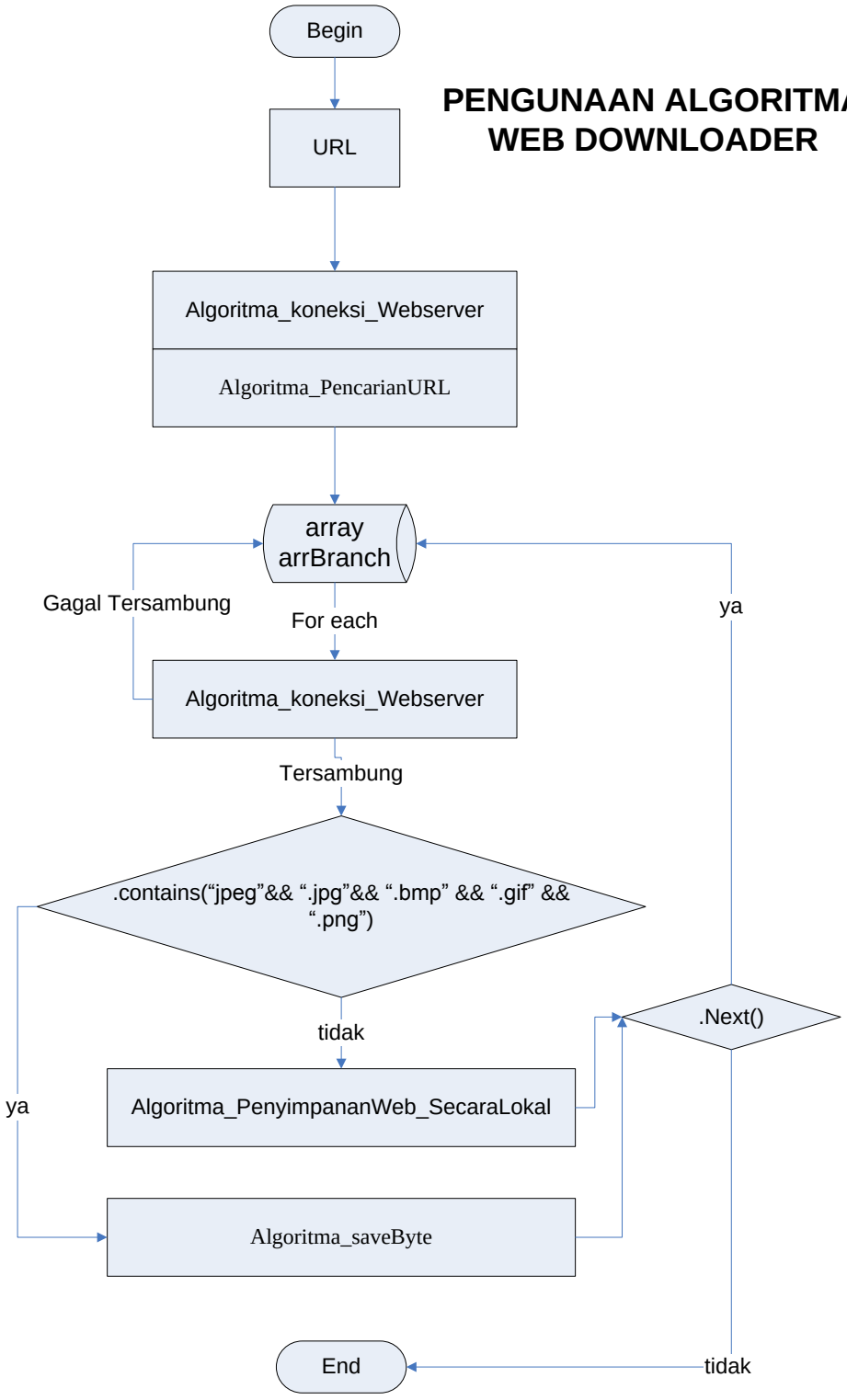
Algoritma perubahan *fixed path* menjadi *relative path* sebagaimana yang dijelaskan pada sub bab 4.3 dapat digambarkan secara detail menggunakan *flowchart* seperti yang tampak pada gambar 4.2



Gambar 4.. Flowchart Algoritma Pencarian URL



Gambar 4. Flowchart Algoritma Mengubah Fixed Path Menjadi Relative Path

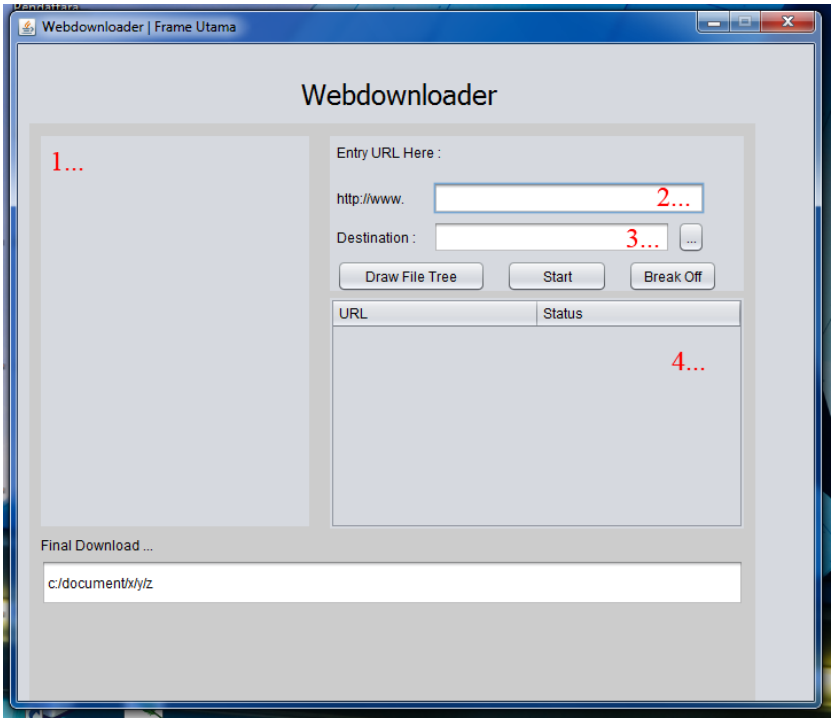


Gambar 4.3 Flowchart Keseluruhan Pengunaan Algoritma Web Downloader

BAB V
IMPLEMENTASI WEBDOWNLOADER

Bab ini menjelaskan mengenai penggunaan daripada aplikasi webdownloader. Aplikasi dapat dijalankan lewat file webdownloader.jar dan dengan cara penggunaan yang benar maka aplikasi dapat digunakan secara benar.

Memberikan penjelasan mengenai hasil implentasi dari algoritma yang telah dijelaskan pada Bab IV, implementasi menggunakan pemrograman Java



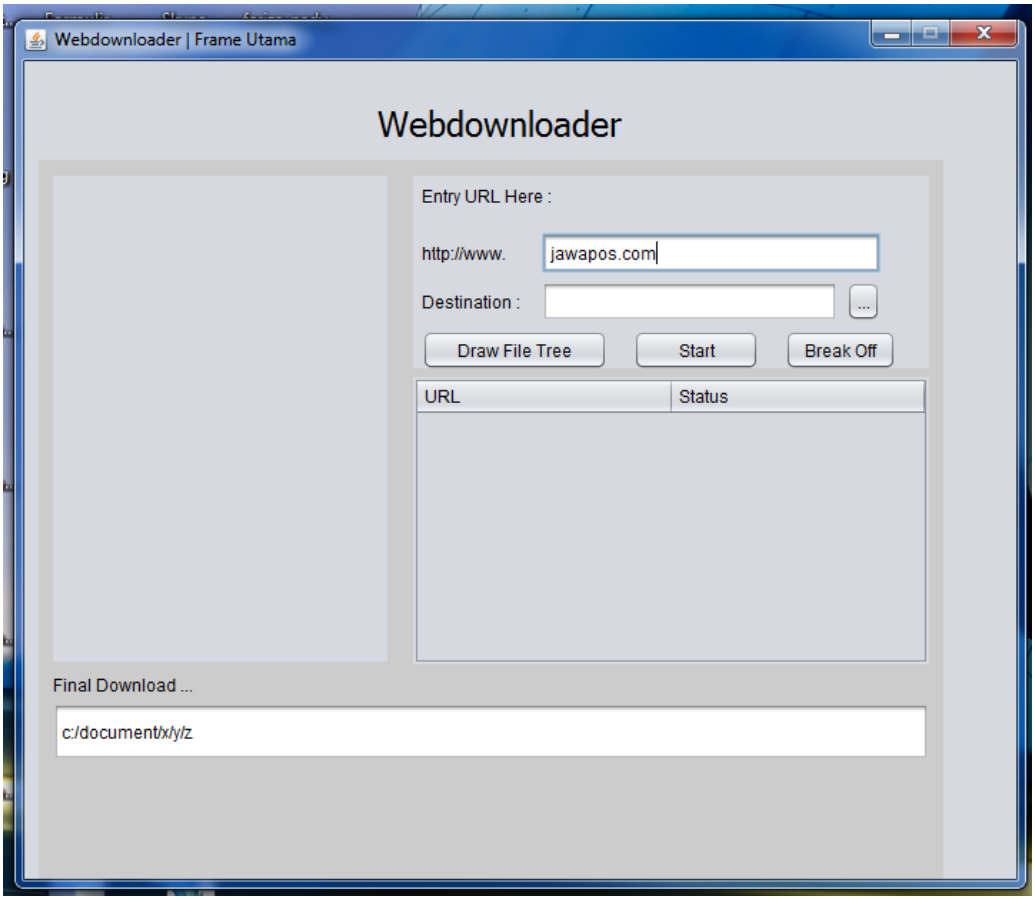
Gambar 5.1. : Interface program Web downloader

Penggunaan aplikasi *Webdownload*er ini cukup mudah, sebab hanya terdapat 4 bagian yang harus diperhatikan oleh user. Bagian tersebut yaitu :

1. Merupakan *File tree* yang menujukan susunan direktori dan file yang dari hasil *download* dari sebuah alamat yang dimasukan pada penujuk 2.

2. Sebuah inputbox yang memungkinkan user memberikan sebuah URL tertentu lewat ketikan keyboard atau screen keyboard.
3. Sebuha *open file dialog* yang memungkina user memilih sendiri tempat dia akan menyimpan project yang barus saja di*download*.
4. Sebuah tabel sederhana yang memperlihatkan perkembangan proses yang dilakukan oleh aplikasi.

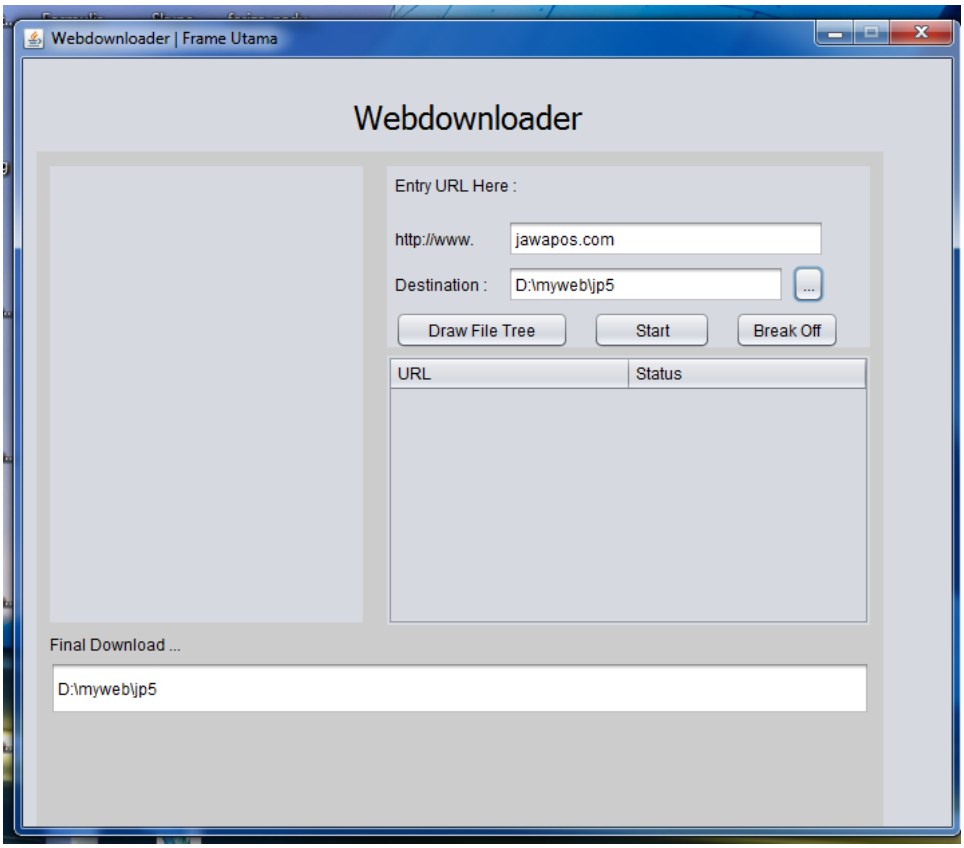
Cara penggunaan Aplikasi ini :



Gambar 5.2 Langkah pertama penggunaan aplikasi

Langkah pertama adalah memberikan alamat URL dari *Website* yang akan di*download*. URL yang diinputkan tidaklah harus diketikan secara lengkap dengan protokol httpnya hanya cukup alamat URL tanpa http://www.

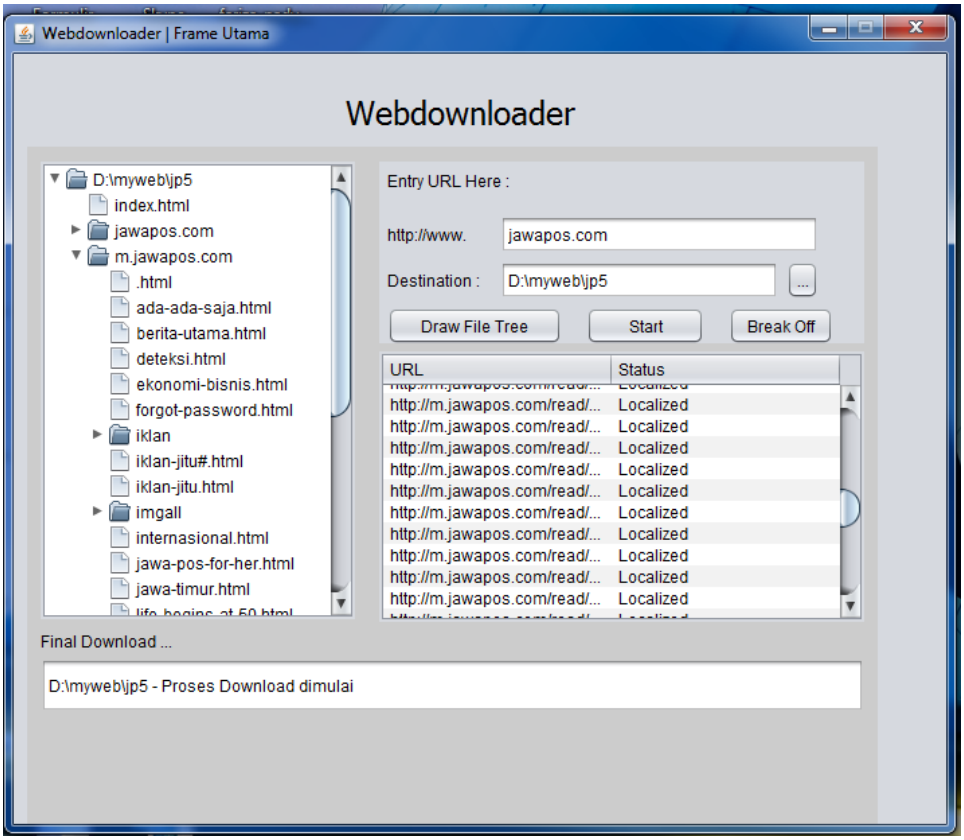
Hal yang patut diperhatikan adalah bisa saja terjadi URL yang diberikan diteruskan (*redirect*) kedalam URL mobile yang jelas mempengaruhi tampilan *Web*site yang dituju. Hal ini disebabkan class URL yang digunakan ketika melakukan koneksi ke *Webserver* dikenali sebagai Java mobile.



Gambar 5.3 Langkah kedua penggunaan aplikasi

Langkah kedua adalah memilih tujuan lokasi direktori penyimpanan dari proyek ini nantinya. Ketika melakukan klik pada tombol yang berlambang "…" maka akan muncul kotak dialog yang menunjukan pembuatan folder tujuan. Jika tidak ingin membuak kotak dialog maka cukup mengetikan *path* sesuai dengan ketentuan pada sistem operasi Windows.

Setelah itu dapat langsung melakukan klik pada tombol start untuk memulai proses *download*. Tombol Break Off digunakan untuk memaksa aplikasi berhenti ditengah jalan, file yang sudah ter*download* tidak akan menghilang meskipun tombol break off ditekan.

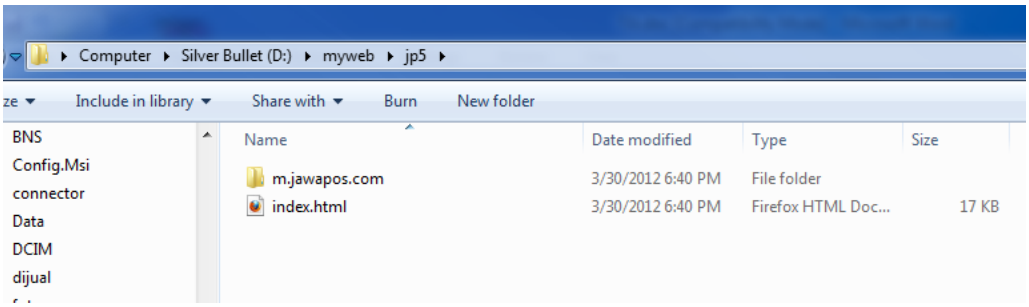


Gambar 5.4 Langkah ketiga penggunaan aplikasi

Langkah ketiga adalah user mampu memperhatikan proses yang dilakukan oleh aplikasi dengan menyaksikan dengan seksama tabel yang terus bertambah dari setiap URL yang sedang dikunjungi dan proses yang dilakukan. Terdapat 2 buah status yang akan muncul :

- Crawl – URL tersebut baru saja ditemukan untuk disimpan
- Finished – URL tersebut telah selesai di*download* dan dibuat menjadi file

Setelah semua proses selesai akan muncul sebuah pesan yang member kabar bahwa semua proses telah selesai dan pada bagian *file tree* di bagian-1 akan menampilkan susunan dari proyek yang baru saja selesai di*download*. Direktori proyek dapat ditemukan dengan cara membuka file explorer dan mencari *path* yang dimasukan pada text box Destination.



Gambar 5.5 Contoh Path Yang Dituju Sesuai Inputan Awal

Seluruh file yang diijinkan oleh *Webserver* telah ter*download* sesuai dengan susunan URL sebenarnya. Dilain sisi jika ada gambar atau obyek tertentu yang tidak muncul dapat disebabkan karena *Webserver* memiliki *setting* khusus yang tidak mengijinkan untuk di*download*.

BAB VI
PENUTUP

Merupakan bab dimana berisi kesimpulan dan saran untuk skripsi ini, menyikapi rumusan masalah dan tujuan penulisan serta hasil implementasi program *web downloader*

5.1 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.

5.2 Saran

Dilain pihak tidak menutupi bahwa ada bagian yang masih dapat diperbaiki oleh karena batasan masalah yang ada juga diberikan pada skripsi kali ini serta juga beberapa ketidak mampuan aplikasi *web downloader* kali ini. Batasan masalah tersebut harus dapat dilewati jika pada masa mendatang tema ini diangkat kembali.

Kekurangan dan batasan yang ada pada penulisan skripsi ini beserta saran yang dianjurkan yaitu sebagai berikut :

1. Tidak dapat melakukan pelacakan atau penyimpanan halaman *web* yang mengandung AJAX. Saran untuk dapat melakukan penyimpanan pada halaman yang mengandung AJAX adalah dengan menanamkan pula sebuah javascript yang memaksa website yang akan di*download* melakukan *request* mandirinya kepada *webserver*, setelah seluruh halaman utuh baru kemudian disimpan ke lokal memori.
2. Sebaiknya aplikasi juga dapat men-*download* halaman web yang bersifat *server side scripting*. Misalkan web server menggunakan PHP maka file PHP dari website tersebut yang didownload. Hal ini dapat dilakukan dengan menelusuri *hole* dari *webserver* atau memotong *response* dari webserver sebelum diubah menjadi HTML *response*.
3. Tidak dapat melakukan penyimpanan terhadap <frame> atau *embed object* yang merupakan sebuah obyek rumit dan biasanya dilindungi oleh *web server* asal. Saran, dapat mengelabuhi web server dengan cara melakukan download seperti yang dilakukan user secara manual. Dilain pihak, kekurangan dari metode ini adalah harus mengetahui ekstensi file dari obyek yang akan di-*download* tersebut.

DAFTAR PUSTAKA

Cahyadi,Daniel. Kiswono Prayogo. **Modul Praktikum Java**. 2011. Surabaya: UPH Surabaya.

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

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

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

Ragged, Dave. Arnaud Le Hors. Ian Jacobs. HTML 40. 1999**.** Europe: W3C

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