II. TEORI DASAR

1. WORLD WIDE WEB

World Wide Web merupakan *framework* arsitektur untuk memasuki dokumen-dokumen yang saling berhubungan yang tersebar di ribuan mesin di seluruh *internet*. *Interface* grafisnya menyebabkan *World Wide Web* menjadi populer sehingga mudah digunakan oleh pemula sekalipun. Di samping itu *World Wide Web* juga menyediakan informasi yang sangat lengkap pada hampir semua masalah.

World Wide Web atau *web* (juga dikenal sebagai WWW) pada dasarnya sebuah sistem *client-server*. Bahasa yang dipakai membuat halaman-halaman *web* antara lain HTML dan Java.

2. CLIENT

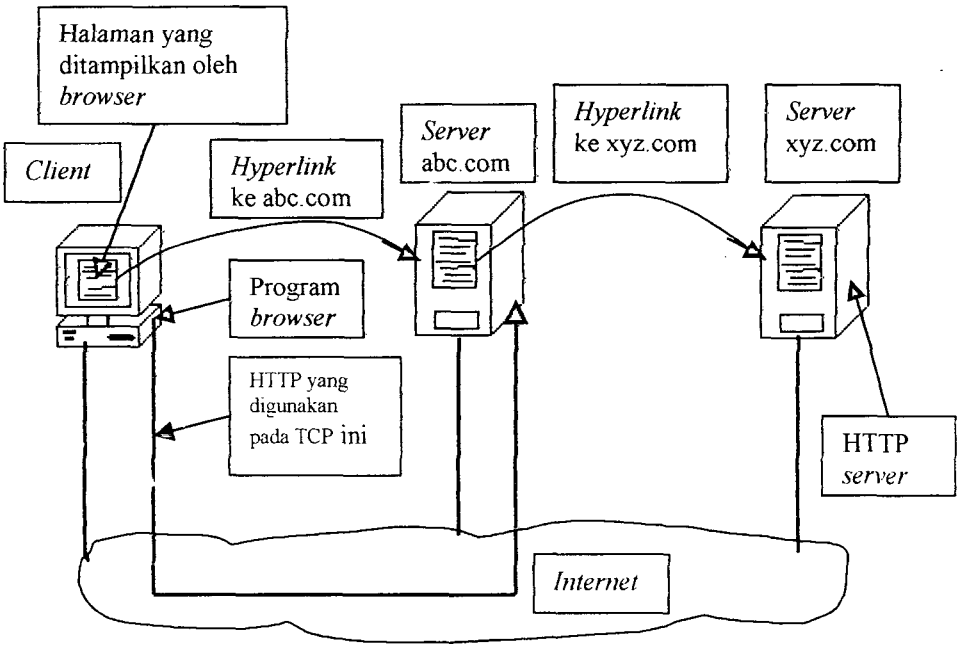
Dari sudut pandang *client*, *web* terdiri dari kumpulan dokumen yang tersebar di seluruh dunia, untuk singkatnya biasanya cukup disebut *page* (halaman). Setiap halaman dapat berisi *link* (pointer) ke halaman lainnya di manapun tempatnya di dunia. *Client* dapat mengikuti *link*-nya (misalnya dengan mengkliknya), yang mana kemudian akan membawa *client* ke halaman yang dituju. Proses ini dapat berulang ulang terus, mungkin dapat melalui ratusan halaman. Halaman yang menunjuk halaman lain disebut *hypertext*.

Halaman dapat dilihat dengan program yang disebut *browser*, di mana Internet Explorer dan Netscape merupakan dua *browser* yang populer. *Browser* mengambil halaman yang diminta, mengintepretasikan teks dan perintah-perintah format yang berada di dalamnya, dan menampilkan halaman yang telah terformat dengan benar di layar. Seperti banyak halaman-halaman *web*, dimulai dengan sebuah judul yang berisi informasi tertentu, dan diakhiri dengan alamat *email* pemilik halaman itu. *String-string* teks yang mempunyai *link* ke halaman lainnya, yang disebut *hyperlink*, ditandai dengan cara tertentu, bisa berupa garis bawah atau warna. Untuk mengikuti sebuah *link*, *client* dapat menempatkan *cursor* pada daerah yang ditandai itu dan kemudian memilihnya dengan cara mengklik tombol *mouse*.

Tidak semua halaman dapat dilihat dengan cara konvesional. Misalnya, beberapa halaman terdiri dari *audio track*, *video clip* atau keduanya. Pada saat halaman *hypertext* dicampur dengan media lainnya, maka akan menghasilkan apa yang disebut dengan *hypermedia*.

3. SERVER

Setiap situs *web* memiliki proses *server* yang mengamati koneksi yang masuk dari *client* pada TCP *port* 80. Setelah koneksi ditetapkan, *client* mengirim *request* dan *server* mengirim jawabannya. Setelah itu koneksi dilepaskan. Gambar 2.1 menunjukkan cara beberapa *web* bekerja sama.



Gambar 2.1

Cara Beberapa Web Bekerja Sama

Sebagai contoh apabila seorang *user* membuka halaman yang memiliki URL `http://www.w3.org/hypertext/WWW/TheProject.html`, maka tahapan yang terjadi sampai ditampilkannya halaman pada layar adalah sebagai berikut :

1. *Browser* menentukan URL.
2. *Browser* meminta DNS bagi alamat IP `www.w3.org`.
3. DNS menjawab dengan 18.23.0.23.
4. *Browser* membuat koneksi TCP ke port 80 di 18.23.0.23.
5. Kemudian *browser* mengirimkan perintah `GET /hypertext/WWW/TheProject.html`.
6. *Server* `www.w3.org` mengirimkan file `TheProject.html`.
7. Koneksi TCP dilepaskan.
8. *Browser* menampilkan seluruh teks di dalam file `TheProject.html`.

9. *Browser* mengambil dan menampilkan seluruh citra yang terdapat di dalam `TheProject.html`.

4. HTML (HYPERTEXT MARKUP LANGUAGE)

HTML merupakan bahasa *markup* (penandaan), yaitu sebuah bahasa untuk menerangkan cara pemberian *format* bagi sebuah dokumen. Istilah *markup* berasal dari *copy editor* dahulu yang memberikan tanda dokumen-dokumen untuk dikirimkan ke *printer*, yang biasanya berisi tentang jenis huruf yang dipakai, dan lain-lain. Jadi bahasa *markup* berisi perintah pemformatan secara eksplisit. Misalnya dalam HTML, <B> berarti tanda mengawali mode cetak tebal dan </B> berarti tanda mengakhiri mode cetak tebal.

Tabel 2.1 merupakan tabel berisi daftar *tag* HTML yang umum digunakan :

Tabel 2.1

Daftar Tag HTML

Tag	Keterangan
<HTML>...</HTML>	Deklarasi bahwa halaman Web ditulis dalam HTML
<HEAD>...</HEAD>	Membatasi kepala halaman
<TITLE>...</TITLE>	Mendefinisikan title (tidak ditampilkan pada halaman)
<BODY>...</BODY>	Membatasi badan halaman
<H*n*>...<H*n*>	Membatasi tingkatan heading *n*
<B>...</B>	Mode cetak tebal
<I>...</I>	Mode cetak miring
<UL>...</UL>	Membatasi daftar tidak berurut (daftar butir)
<OL>...</OL>	Membatasi daftar berurut
<MENU>...</MENU>	Membatasi menu item-item <LI>
<LI>	Mengawali item list <tidak terdapat </LI>
 	Pindah baris
<P>	Mengawali sebuah paragraf
<HR>	Garis horisontal
<PRE>...</PRE>	Teks preformat; tidak diformat ulang
<IMG SRC="...">	Mendefinisikan gambar
<A HREF="...>...</A>	Mendefinisikan sebuah hyperlink

Halaman *web* yang terdiri dari kepala dan badan diapit oleh *tag* <HTML> dan </HTML>. Sedangkan bagian kepala sendiri diapit oleh *tag* <HEAD> dan </HEAD>. Untuk bagian badan diapit oleh tag <BODY> dan </BODY>. Perintah-perintah yang berada dalam *tag* tersebut disebut *directive*. *Tag* dapat ditulis dengan huruf besar maupun huruf kecil. Jadi <HEAD> dan <head> akan sama artinya, namun *tag* yang penulisannya dengan huruf besar lebih memudahkan untuk dilihat.

Item utama dalam *head* adalah judul, yang dibatasi oleh <TITLE> dan </TITLE>. Jenis informasi tertentu dapat dicantumkan di sini. Namun informasi itu sendiri tidak ditampilkan di layar.

Heading dibuat oleh *tag* <Hn>, dimana *n* merupakan bilangan dalam range 1 sampai dengan 6. <H1> merupakan *heading* yang paling penting, sedangkan <H6> adalah *heading* yang paling tidak penting. Tampilan *heading-heading* ini di layar tergantung pada *browser*. Umumnya *heading* dengan nomor kecil akan ditampilkan dalam bentuk huruf yang lebih besar dan tebal. *Browser* dapat juga memilih menggunakan warna-warna yang berlainan bagi setiap tingkatan *heading*.

Tag <B> dan <I> digunakan untuk membuat mode cetak tebal dan cetak miring. Selain menspesifikasikan *sytle* secara fisik seperti cetak tebal dan cetak miring, pembuat dapat juga menggunakan *sytle logic* seperti <DN> (*define*), <EM> (*weak emphasis*), <STRONG> (*strong emphasis*), dan <VAR> (*program variable*). *Sytle logic* ditentukan di dalam sebuah dokumen yang disebut *sytle sheet*. Keuntungan pemakaian *sytle logic* dengan mengubah

sebuah definisi, maka semua variabel dapat diubah, misalnya dari cetak miring menjadi huruf yang lebar.

HTML menyediakan berbagai mekanisme untuk membuat *list*, termasuk *list* bersarang (*nested list*). *Tag* <UL> mengawali daftar tanpa urutan. *Item-item* individual, yang ditandai dengan *tag* <LI> di dalam sumber, akan muncul dengan tanda bulatan (•) di depannya. Varian mekanisme ini adalah <OL>, yang dimaksudkan untuk daftar yang berurutan. Ketika *tag* ini digunakan, *item-item* <LI> akan diberi nomor oleh *browser.* Pilihan ketiga adalah <MENU>, yang biasanya menghasilkan daftar yang lebih padat di layar, tanpa bulatan dan tanpa nomor.

Tag <DIR> digunakan membuat tabel pendek. <DL> dan </DL> dapat membuat daftar definisi (ringkasan) dengan dua bagian *entry*, yang masing-masing bagiannya ditentukan oleh <DT> dan <DD>. Yang pertama ditujukan untuk nama, sedangkan *tag* yang kedua untuk isinya.

Semua *tag*
, <P>, dan <HR> menandai batas antara dua daerah teks. *Format* yang tepat dapat ditentukan oleh *sytle sheet* yang berkaitan dengan halaman. *Tag*
 memaksa untuk pindah baris. Umumnya *browser* tidak menyisipkan baris baru setelah
. Sebaliknya <P> mengawali sebuah paragaf baru, yang dapat menyisipkan baris kosong dan berbagai indentitasi.

HTML memungkinkan *image* untuk dimasukkan secara *in-line* pada halaman *web*. *Tag* <IMG> menyatakan bahwa *image* dimasukkan ke posisi itu pada halaman. *Tag* ini memiliki beberapa *parameter*. *Parameter* SRC menentukan URL dari *image*. *Parameter* ALIGN yang mengontrol perataan *image* terhadap baris bawah teks (TOP, MIDDLE, BOTTOM). *Parameter*

ALT yang menyediakan teks untuk dipakai menggantikan *image* pada saat pengguna mematikan fungsi *image*. Dan *parameter* ISMAP yang merupakan sebuah *flag* yang mengindikasikan bahwa *image* merupakan sebuah *map* yang aktif. Seperti halnya dengan <IMG>, <A> memiliki bermacam-macam *parameter*, termasuknya di antaranya HREF (URL), NAME (nama *hyperlink*), dan METHODS (metode *access*). Teks yang terdapat di antara <A> dan </A> akan ditampilkan . Bila *tag* itu dipilih, maka *hyperlink* akan menuju halaman berikutnya. Dalam hal ini diizinkan juga untuk menaruh *image* <IMG>, dimana pengklikan pada citra akan mengaktifkan *hyperlink*.

Tag <CAPTION> dapat digunakan untuk memberikan judul gambar. Setiap baris diawali dengan *tag* <TR> (*Table Row*). Sel-sel individu ditandai dengan <TH> (*Table Header*) atau <TD> (*Table Data*).

Dan yang terakhir *browser* dapat memainkan file video AVI dan MPG yang ditulis dalam bahasa HTML. Sintaks penulisannya adalah :

```
<IMG DYNSCR="nama_file.avi/.mpg" START"FILE OPEN"
WIDTH="200" HEIGHT="100">
```

Supaya *browser* dapat memainkan file video DAT, maka file DAT tersebut harus dikonversikan terlebih dahulu ke AVI atau MPG menggunakan konversi video, seperti : iFilm Edit atau VCD Cutter.

5. CGI MENGGUNAKAN PERL

Common Gateway interface (CGI) adalah suatu *interface* yang menghubungkan *web server* dengan *user* yang sedang mengakses *site* tersebut agar dapat terjadi interaksi antara keduanya. Salah satu contoh interaksi ini

adalah *form* yang ditulis dalam bahasa HTML. *Form* HTML merupakan suatu metode yang digunakan untuk pengiriman data melalui jaringan *internet*.

Karena CGI merupakan *common interface*, maka harus diimplementasikan menggunakan suatu bahasa pemrograman. Tidak ada batasan mengenai bahasa pemrograman apa yang dapat digunakan untuk mengimplementasikan CGI, asal bahasa pemrograman tersebut dapat melakukan tiga hal di bawah ini :

1. Mencetak ke *standard output*.
2. Membaca dari *standard input*.
3. Membaca dari *environment variable*.

Dua bahasa pemrograman yang umum digunakan untuk pemrograman CGI adalah C dan PERL. Untuk *form* dalam program ini dibuat menggunakan bahasa pemrograman PERL versi 5.

Practical Extraction Report Language (PERL) merupakan suatu bahasa pemrograman yang lebih banyak digunakan untuk mengolah teks dan berstruktur *script*. PERL tersedia dalam banyak *platform*, mulai dari DOS hingga UNIX. Sampai saat ini, PERL sudah mencapai versi 5. PERL adalah produk GNU, dan pertama kali dibuat oleh Larry Wall.

Supaya data yang dikirimkan dapat berguna bagi CGI maka data tersebut harus dilakukan *parsing*. PERL telah memiliki *library* standar untuk *parsing* yang ditulis oleh Steve Brenner, yaitu : cgi-lib.pl. Untuk menggunakannya di dalam program kita cukup kita tulis : `require "cgi-lib.pl";`. Dan fungsi untuk membaca data yang didapat dari formulir adalah `&ReadParse()`.

6. JAVASCRIPT

Selain CGI, ada bahasa pemrograman lain yang juga digunakan untuk meningkatkan interaktivitas halaman *web*, yaitu Java. Java dikembangkan oleh Sun Microsystems. Sayangnya Java sangat sulit dipelajari oleh pemula, terutama yang masih asing dengan pemrograman C atau C++. Oleh karena itu Netscape Communications bersama dengan Sun Microsystems mengembangkan bahasa *script* yang diberi nama JavaScript.

Dengan JavaScript kita dapat membuat suatu halaman *web* menjadi interaktif dan juga cerdas. Sebagai contoh penggunaan JavaScript adalah untuk melakukan pengecekan sah atau tidaknya terhadap data yang di-*input*-kan oleh *user* sebelum data tersebut masuk ke *server*.

Program JavaScript dituliskan pada file HTML dengan menggunakan *tag* `<SCRIPT>`. Supaya *browser* dapat mengenali *script* yang kita tulis, maka dalam file HTML tersebut harus bersintaks :

```
<SCRIPT LANGUAGE="JavaScript">
// Program JavaScript ditulis disini
</SCRIPT>
```

Di dalam *form* pada program *Video On Demand* tersebut sebagian juga menggunakan JavaScript. JavaScript pada *form* dalam program *Video On Demand* ini digunakan untuk mengecek apakah semua *input* yang dimasukkan oleh *user* sudah diisi lengkap atau belum dan juga apakah *input Password* dan *Confirm Password* sama atau tidak.

Untuk melakukan pengecekan tersebut cukup dengan pernyataan `if`. Sintaks pernyataan `if` adalah :

```
if (kondisi)
{
//pernyataan-pernyataan yang dieksekusi jika kondisi
//terpenuhi
}
```

7. JAVA

HTML mampu mendeskripsikan halaman *web* yang statis, termasuk tabel dan gambar. Tetapi dengan Java, seperti yang sudah dijelaskan sebelumnya, kita mampu membuat halaman *web* tersebut menjadi lebih interaktif.

7.1 Bahasa Pemrograman Java

Gagasan utama penggunaan Java untuk halaman *web* adalah bahwa halaman *web* dapat menunjuk ke program Java kecil, atau yang biasa disebut *applet*. Ketika *browser* menemukan *applet*, maka *applet* tersebut diambil ke mesin *client* dan dieksekusi disana secara aman.

Mekanisme sistem dari bahasa Java memiliki tiga bagian :

1. Kompiler Java ke *bytecode*.
2. *Browser* yang mengerti *applet*.
3. *Interpreter bytecode*.

Developer menulis *applet* dengan Java, kemudian mengkompilasinya ke dalam *bytecode*. Untuk memasukkan *applet* yang telah dikompilasi ke halaman *web*, maka telah dibuat *tag* HTML yang baru `<APPLET>`. Sintaks secara umum adalah :

```
<APPLET          CODE=nama_file.class          WIDTH=100
HEIGHT=200></APPLET>
```

Ketika mengetahui adanya *tag* <APPLET>, maka *browser* mengambil *applet* nama_file.class yang telah dikompilasi dari situs halaman *web* saat itu. *Browser* kemudian meneruskan *applet* tersebut ke *interpreter bytecode* lokal untuk dieksekusi. *Parameter* WIDTH dan HEIGHT memberikan ukuran *default applet* pada windows, dalam *pixel.*

Dalam skala kecil bahasa pemrograman Java mirip dengan C dan C++. Java memiliki delapan jenis data primitif, seperti ditunjukkan pada tabel 2.2.

Tabel 2.2

Tipe Data Dasar Java

Jenis	Ukuran	Keterangan
Byte	1 byte	Integer bertanda antara –128 sampai +127
Short	2 byte	Integer 2 byte bertanda
Int	4 byte	Integer 4 byte bertanda
Long	8 byte	Integer 8 byte bertanda
Float	4 byte	Bilangan floating point IEEE 4 byte
Double	8 byte	Bilangan floating point IEEE 8 byte
Boolean	1 bit	Nilai benar dan salah
Char	2 byte	Karakter di dalam unicode

Java memiliki beberapa pernyataan yang sintaksnya sama dengan C dan C++. Beberapa pertanyaan Java ditunjukkan dalam tabel 2.3.

Tabel 2.3

Jenis Pernyataan Java

Pernyataan	Keterangan
Assignment	Pemberian nilai
If	Pilihan Boolean
Switch	Memilih case
For	Iterasi
While	Perulangan
Do	Perulangan
Break	Pernyataan keluar
Return	Mengembalikan nilai
Continue	Perulangan berikutnya
Throw	Mencapai pengecualian
Try	Pembuatan lingkup
Synchronized	Eksklusi mutual

Sembilan pernyataan pertama memiliki sintaks dan sematik yang sama seperti bahasa C, kecuali pada ekspresi `Boolean`. Di samping itu, pernyataan `Break` dan `Continue` dapat mengambil label yang mengindikasikan label mana yang membentuk *loop* ke luar atau perulangan.

Dua pernyataan berikutnya adalah pernyataan dalam bahasa C++, bukan bahasa C. Pernyataan `Throw` dan `Try` berkaitan dengan *Exception* Java yang mendefinisikan bermacam-macam *standard exception*, seperti pembagian dengan nol, dan mengizinkan pemrogram untuk mendefinisikan dan mencapai *exception* yang dibuatnya sendiri. Pemrogram dapat menulis *handler* untuk menangkap *exception*, yang membuatnya tidak perlu lagi mencoba secara terus-menerus bila terjadi suatu kesalahan. Pernyataan `throw` mencapai *exception*, dan pernyataan `try` mendefinisikan lingkup yang berkaitan dengan *exception handler* dengan memakai blok kode dimana kemungkinan *exception* dapat terjadi.

Pernyataan `synchronized` merupakan sesuatu yang baru bagi Java dan harus menangani dengan adanya kenyataan bahwa program Java dapat memiliki sejumlah `Thread` kontrol. Untuk menghindarkan *race condition*, pernyataan ini digunakan untuk membatasi blok kode yang tidak boleh memiliki lebih dari satu `Thread` aktif dalam satu kesempatan yang sama. Blok-blok kode seperti itu biasanya disebut daerah kritis. Ketika pernyataan `synchronized` dieksekusi, `thread` yang mengeksekusinya harus memperoleh kunci yang berkaitan dengan daerah kritis, mengeksekusi kode, dan melepaskan kunci. Bila kunci tidak dapat diperoleh, `thread` menunggu sampai kunci itu berada dalam keadaan bebas. Dengan memadu keseluruhan *procedure* seperti itu dan menggunakan variabel-variabel kondisi, maka pemrogram akan memiliki kemampuan untuk mengawasi.

Di samping keduanya berbentuk bahasa berorientasi objek berbasis C, tetapi Java dan C++ memiliki beberapa perbedaan. Beberapa fitur-fitur telah dihilangkan dari Java untuk alasan keamanan atau memudahkan pembacaan. Fitur-fitur yang dihilangkan ini meliputi *#define, typedef, enums, unions, structs, operator overloading,* pointer eksplisit, variabel global, fungsi yang berdiri sendiri, dan *friend function.* Terdapat juga fitur-fitur yang ditambahkan untuk memungkinkan Java lebih kaya. Fitur yang ditambahkan meliputi *garbage collection, multithreading, interface object,* dan *package*[1].

---

[1] Andrew S. Tanenbaum. *Jaringan Komputer Jilid 2 Edisi 3.* (Jakarta: Prehallindo, 1997). p. 275.

7.2 Multimedia Java

Multimedia yang dimiliki Java Development Kit v1.1 meliputi : *image*, animasi, dan audio. Java Development Kit v1.1 belum mendukung untuk multimedia video.

Pada Java Development Kit v1.1 hanya mendukung file audio AU (format file audio yang dimiliki oleh Sun). Agar dapat menggunakan format-format suara yang lain seperti : WAVE (format file audio milik Microsoft Windows), AIFF (The Audio Interchange File Format yang biasa digunakan oleh Macintosh dan Silicon Graphics computers), RMF (The Rich Music Format yang dibuat oleh Headspace, Inc.), MIDI (The Musical Instrument Digital Interface), maka harus dikonversikan terlebih dahulu menggunakan program suara yang mampu menangani konversi audio. Apabila tetap dicoba menggunakan file audio selain AU maka *class* `sun.audio.NativeAudioStream` akan melontarkan `InvalidAudioFormatException`. *Exception* ini tidak akan menghentikan jalannya *applet*, hanya saja file audio tersebut tidak dimainkan. File AU merupakan file audio 8 bit, 8.000 Hz dan menggunakan format kompresi *μlaw*.

Ada dua cara untuk memainkan file suara dari dalam *applet* :

1. *Method* `play()`.

   Method ini didefinisikan sebagai berikut :

   a. `public void play(URL url)` dimana memainkan klip audio yang ditunjuk oleh URL.

b. `public void play(URL url, String name)` dimana memainkan klip audio yang ditunjuk oleh URL dan nama file audio.

Method `play()` memiliki satu kekurangan. Saat pertama kali memanggil *method* `play()` menggunakan parameter file audio, dan apabila file audio tersebut belum pernah digunakan sebelumnya, maka *method* ini harus men-*download*-nya terlebih dahulu sehingga mempengaruhi responsivitas.

2. *Method* `getAudioClip()`.

Method ini didefinisikan sebagai berikut :

a. `public AudioClip getAudioClip(URL url)` dimana memainkan klip audio yang ditunjuk oleh URL.

b. `public AudioClip getAudioClip(Url url, String name)` dimana memainkan klip audio yang ditunjuk oleh URL dan nama file audio.

Method `getAudioClip()` ini digunakan untuk memainkan klip audio yang berulang-ulang tanpa mempengaruhi responsivitas. Ketika `getAudioClip()` ini dijalankan maka terdapat tiga *method* yang dapat dipanggil : `Play`, `Loop`, dan `Stop`. *Method* `Play` untuk memainkan file audio hanya sekali. *Method* `Loop` untuk mengulang-ulang file audio secara berlanjut. *Method* `Stop` untuk menghentikan file audio baik yang dimainkan oleh *method* `Play` atau *method* `Loop`.

Kedua *method* di atas termasuk di dalam *package* `java.applet` dan termasuk di dalam *class* `java.applet.Applet`. Yang termasuk *public member* dari `java.applet.Applet` adalah :

```
public class Applet extends Panel {
   public final void setStub(AppletStub stub)
   public boolean isActive()
   public URL getDocumentBase()
   public URL getCodeBase()
   public String getParameter(String name)
   public AppletContext getAppletContext()
   public void resize(int width, int height)
   public void resize(Dimension d)
   public void showStatus(String msg)
   public Image getImage(URL url)
   public Image getImage(URL url, String name)
   public AudioClip getAudioClip(URL url)
   public AudioClip getAudioClip(URL url, String
      name)
   public String getAppletInfo()
   public String[][] getParameterInfo()
   public void play(URL url)
   public void play(URL url, String name)
   public void init()
   public void start()
   public void stop()
   public void destroy()
}
```