Di Sini Java, di Sana Java

Ternyata nama Java tidak hanya milik orang Jawa atau Indonesia. Saatnya Anda ikut merasakan kehebatan bahasa pemrograman Java, yang dapat berjalan di berbagai sistem operasi dan peranti keras komputer.

Java kini merajai komputasi di sistem *enterprise* dan Internet. Padahal saat dirintis sekitar tahun 90-an, Java justru dirancang buat sistem kecil-kecil seperti TV kabel atau *home theater*. Setelah berjalan lebih dari empat tahun, ternyata pasar tersebut tidak berkembang. Pemimpin proyek Java, **James Gosling**, bahkan sampai khawatir masa depannya bakal suram. Lalu terbersit ide, mengapa tidak mencoba ke Internet? Keputusan setengah putus asa di tahun 1994 ini ternyata menjadi nasib baik mereka dan juga seluruh dunia.

Artikel ini secara singkat memperkenalkan keunikan dan kelebihan Java yang membuatnya sukses. Setelah itu akan dibahas bagaimana menyiapkan sarana untuk membangun dan menjalankan program Java di mesin Linux Anda.

Sekali tulis, lari di mana saja

Ada tiga ide cemerlang yang menjadi ciri kesuksesan Java, yaitu bahasa, kode byte, dan *virtual machine*. Seperti terlihat pada Gambar 1, siklus hidup Java dimulai dari kode sumber yang ditulis manusia (pemrogram) memakai bahasa Java. Bahasa ini merupakan bahasa berorientasi objek yang diturunkan dari C++ dengan banyak penyempurnaan. Pada umumnya, para pakar berpendapat bahwa bahasa Java memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk diimplementasikan. Kini universitas-universitas di berbagai negara berpaling dari Pascal atau C++ dan memilih Java sebagai bahasa untuk belajar memprogram.

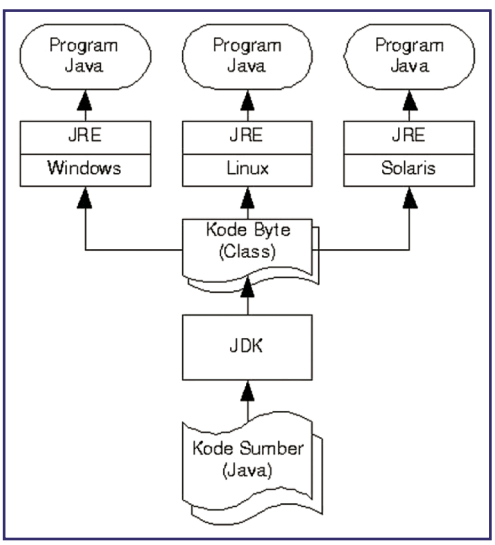
Setelah selesai ditulis, kode sumber Java harus diubah menjadi kode siap eksekusi dengan menggunakan *Java Development Kit* (JDK). Di sini letak keunikan Java. Java menggunakan kode byte yang **portabel** dan **modular**. Portabel karena dia bukan kode mesin prosesor (peranti keras) tertentu, justru sebaliknya dia bisa dimuat ke berbagai landasan komputer maupun sistem operasi. Dia juga modular karena tiap objek dikompilasi menjadi satu file kelas (*class*) yang mandiri. Aplikasi lengkap Java merupakan kumpulan beberapa file kelas. File-file kelas ini dapat disatukan dan dipadatkan menjadi file jar (*Java archive*).

Pada akhirnya, kode byte tersebut akan dijalankan sebagai program oleh *Java Runtime Environment* (JRE). Untuk masing-masing landasan komputer dan sistem operasi, tersedia JRE yang berbeda. JRE inilah yang menyembunyikan si landasan dan menyediakan lingkungan yang serupa bagi program Java agar dapat bekerja sebagai mana mestinya. Dengan strategi ini, Java mampu menjadi peranti lunak yang *"write once run everywhere"*.

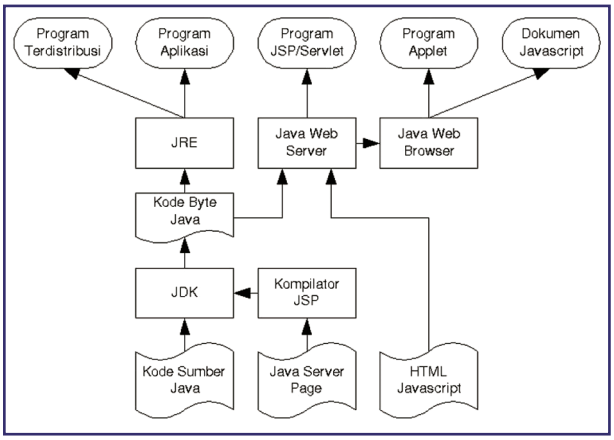
Satu alat buat segala keperluan

Kunci lain kesuksesan Java adalah beragamnya arsitektur program yang didukungnya (Gambar 2). Dengan Java Anda bisa membangun program aplikasi yang berjalan di satu komputer, program terdistribusi yang bekerja antara beberapa komputer, bahkan program *mobile-agent* yang bisa melompat-lompat dari satu komputer ke komputer lainnya di Internet. Arsitektur-arsitektur tesebut sangat menunjang komputasi tersebar dan paralel, sehingga Java sangat populer di kalangan peneliti dan akademik. Sementara itu di dunia bisnis, arsitektur Java yang menarik adalah *applet, servlet, Java Server Page* (JSP), dan *Javascript*. Sejauh ini, arsitektur-arsitektur tersebut merupakan pilihan terlengkap yang bisa disediakan oleh satu alat pengembangan.

Aplikasi adalah arsitektur Java paling klasik, di mana kode sumber dikompilasi menjadi kode byte lalu dijalankan menjadi program di sebuah komputer. Aplikasi Java mampu bekerja dengan antarmuka teks maupun grafik untuk mengerjakan



Gambar 1. Tahap-tahap pengembangan Java



Gambar 2. Berbagai arsitektur Java

segala jenis komputasi dari pengolah numerik, kata, hingga grafik. Dari luar, aplikasi Java tidak ada bedanya dengan aplikasi C++ atau lainnya.

Saat ini, arsitektur Java yang paling bagus pemasarannya adalah yang berhubungan dengan *world wide web* (WWW). Bahkan boleh dikata, Java ikut meledakkan kepopuleran WWW dengan Applet yang mampu menampilkan grafik animatif di jendela *web browser. Web server* men-*download* kode byte applet dari web server, kemudian mengeksekusinya dengan JVM. Setiap applet akan diberi satu area terbatas di layar web browser untuk menampilkan grafik dan masukan papan kunci maupun *mouse*. Dengan demikian, applet mampu mengerjakan komputasi interaktif apa saja mulai animasi sederhana sampai *games*.

Sementara itu untuk keperluan yang ringan-ringan, Netscape mengembangkan Javascript berupa *scripting* yang disisipkan di HTML untuk dijalankan langsung oleh web browser. Bahasa Javascript lebih sederhana dari pada bahasa Java serta objek-objeknya hanya khusus untuk memanipulasi dokumen HTML dan lingkungan web browser. Dengan menggabungkan HTML, Javascript ditambah *cascading style sheet (CSS)*, Anda bisa membuat halaman web yang dinamik dengan tata letak menarik.

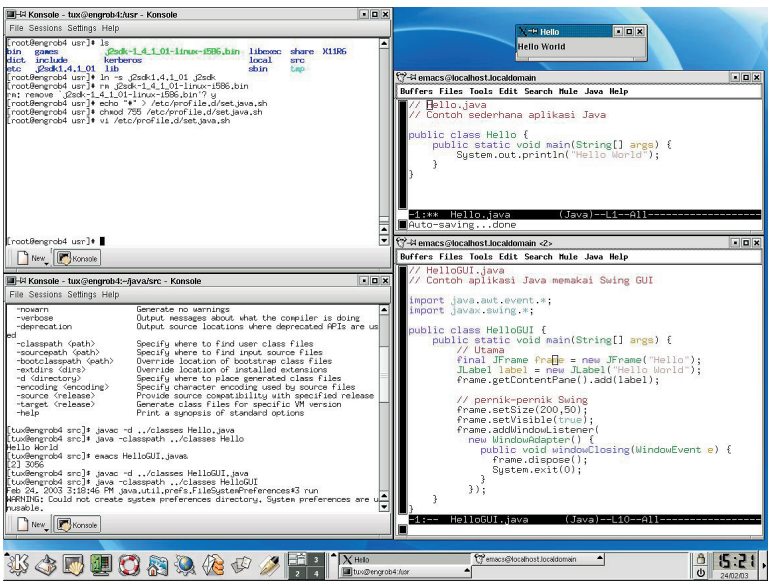
Setelah makin populer, informasi yang tersimpan di web server makin menumpuk sehingga tidak mungkin lagi diedit satu per satu. Karenanya, diperlukan cara otomatis untuk menerbitkan halaman HTML dari informasi mentah misalnya dari database. Arsitektur Java yang dirancang untuk itu adalah servlet dan Java Server Page (JSP).

Kebalikan dari applet, servlet adalah program Java mini yang dijalankan sebagai modul web server. Ketika suatu web browser meminta halaman servlet, web browser akan memakai JVM untuk menjalankan kode byte servlet yang bersangkutan. Servlet mampu mengambil data dari berbagai sumber, mengolahnya, kemudian menghasilkan dokumen HTML yang seterusnya dikirim ke web browser.

Jika di sisi web browser applet ditemani Javascript, maka di sisi server servlet ditemani JSP. Dengan JSP, Anda bisa menyisipkan script bahasa Java dalam dokumen HTML. Saat web browser meminta sebuah dokumen JSP, web browser menerjemahkannya menjadi kode byte, kemudian dieksekusi sebagai jsp-servlet dalam web server. Keluaran jsp-servlet ini berupa halaman HTML, tanpa script Java lagi, yang dikembalikan ke web browser. Dalam praktiknya, JSP sering dipakai sebagai latar depan (*front-end*) yang bekerja sama dengan servlet.

Terlihat bahwa arsitektur-arsitektur Java ini sangat beragam dan sampai saat ini tidak ada teknologi lain yang menyamainya. Tabel 1. memperlihatkan perbandingan Java dengan beberapa bahasa populer lain.

Gambar 3. Pemasangan dan pemrograman Java di Linux



Memasang Java SDK di Linux

Berikut ini kita akan menyiapkan JDK dan JRE Java di Linux agar dapat digunakan membangun dan menjalankan aplikasi Java. Tahap-tahap pemasangan disampaikan dalam bentuk *bash script* (saya ambil dari *history*, jika perlu dapat Anda jalankan. Jika ada file yang harus diedit akan disampaikan dalam kotak terpisah. Dalam melakukan instalasi gunakan *user level root*, sedangkan untuk mencoba pemrograman gunakan user biasa. Seluruh instalasi dan pemrograman dapat dilakukan dari *desktop* GUI (Gambar 3).

Saat ini JDK dan JRE untuk Linux dapat anda peroleh dari empat sumber utama:

- Sun microsystems, merupakan pembuat Java panutan. Sun mengeluarkan tiga kelas paket Java, yaitu J2-SE JRE (hanya berisi JRE), J2-SE SDK (berisi JDK + JRE), dan J2-EE SDK (berisi JDK+JRE dan tools untuk aplikasi enterprise). Versi SE (Standard edition) tersedia gratis dari *http://java.sun.com*.
- Blackdown, merupakan proyek mandiri yang memindah J2-SE Sun khusus ke Linux. *Homepage*-nya di *http://www.blackdown.org*.
- IBM, menawarkan paket Java Developer Kit yang performanya jauh lebih bagus dibanding Sun dan Blackdown, namun versinya agak ketinggalan. (*http://www.ibm.com/java*).
- Kaffe, adalah open source JVM dan pustaka kelas Java. Sayangnya tidak begitu kompatibel (*http://www.kaffe.org*).

CD-ROM *InfoLINUX* edisi April 2003 menyertakan Blackdown J2 SDK versi 1.4.1-01. Ini sama dengan versi terbaru dari Sun, dengan perbaikan dan penyempurnaan khusus untuk Linux.

Sebagai langkah awal, siapkan mesin Linux dengan spesifikasi minimum sebagai berikut:

- Komputer PC Pentium, Memori 128MB, ruang harddisk 500MB.
- Blackdown tidak meminta distro Linux tertentu, tapi Sun menyarankan RedHat 7.1 - 7.3.
- Peket kernel-2.4.18, pustaka glibc-2.2.5, kompiler gcc-3.2. Anda bisa periksa hal ini dengan:

```
rpm –q <nama-paket-tanpa-versi>
```

- X-Free versi 4.1. Anda bisa pakai *desktop manager* apa saja baik KDE, GNOME atau lainnya. Anda juga perlu terminal dan editor teks, misalnya vim atau emacs.

Setelah hal tersebut siap, masukkan CD-ROM *InfoLINUX* edisi April 2003, maka kita bisa mulai memasangan J2 SDK seperti tahap-tahap di Listing-1. Anda juga perlu mengedit skrip Bash seperti Listing-2. Pekerjaan ini harus dilakukan sebagai root.

```
# Listing-1. Perintah-perintah untuk pemasangan j2sdk di Linux

# Buat direktori /usr/java (anjuran Sun) untuk pemasangan
mkdir -p /usr/java
cd /usr/java

# Mount cd-rom, dan salin paket j2-sdk lalu diberi mode bisa-dieksekusi
mount /dev/cdrom /mnt/cdrom
cp /mnt/cdrom/java/j2sdk-1.4.1-01-linux-i586.gcc3.2.bin  /usr/java
chmod a+x j2sdk-1.4.1-01-linux-i586.gcc3.2.bin

# Eksekusi paket tersebut untuk memasangnya.
./j2sdk-1.4.1-01-linux-i586.gcc3.2.bin

# Buat link agar mudah diakses, lalu hapus file instalasi dan umount cdrom
ln -s j2sdk-1.4.1-01 j2sdk
rm j2sdk-*.bin
umount /mnt/cdrom

# Buat bash script untuk mensetup environment yang diperlukan
echo "#" > /etc/profile.d/setjava.sh
chmod 755 /etc/profile.d/setjava.sh
vi /etc/profile.d/setjava.sh

#!/bin/bash

# Listing-2.: /etc/profile.d/setjava.sh
# Bash login script untuk menyiapkan environment Java

# Tempat j2sdk dan variabel-variabel yang diperlukan Java
JAVA_HOME=/usr/java/j2sdk
CLASSPATH=./
PATH=$JAVA_HOME/bin:$PATH

export JAVA_HOME CLASSPATH  PATH
```

■ Tabel 1.

Bahasa/Alat pengembangan	Arsitektur Program			
	Modul Web Server	Scripting Web Server	Modul Web Browser	Scripting Web Browser
Java	servlet	JSP	Applet	Javascript
C++	CGI exe		ActiveX*	
Perl	CGI script			
Phyton	CGI script			
PHP		PHP script		
Visual Basic		ASP*	ActiveX*	VB Script*

*) Hanya di landasan Windows, tidak bisa di Linux.

Membuat aplikasi Java

Sekarang marilah kita mencoba menggunakan Java untuk membuat program aplikasi sederhana. *Login*-lah sebagai user biasa di desktop (bukan di konsole teks). Jalankan terminal untuk mengedit, meng-*kompile* dan mengeksekusi sepeti Listing-3. Silakan pakai editor apa saja, contoh menggunakan emacs,

untuk mengedit kode sumber Java seperti pada Listing-4!

```
# Listing-3. Perintah –perintah membangun dan
menjalankan aplikasi Java

# Membuat direktori kerja di home
cd $HOME
mkdir -p java/src
mkdir -p java/classes
cd java/src

# Membuat kode sumber Java
emacs Hello.java &

# Mengkompile kode sumber java menjadi kode byte,
ditaruh di direktori classes
javac -d ../classes Hello.java

# Menjalankan program Hello
java -classpath ../classes Hello

// Listing-4. Hello.Java
// Contoh tersederhana aplikasi Java

public class Hello {
   public static void main(String[] args) {
      System.out.println("Hello World");
   }
}
```

Membuat aplikasi Java GUI

Java dengan mudah dipakai untuk membuat aplikasi berbasis GUI. Silahkan edit contoh kode sumber pada Listing-6, lalu kompile dan jalankan dengan perintah-perintah seperti pada Listing-5.

```
# Listing-5. Perintah-perintah membuat aplikasi Java GUI
emacs HelloGui.java &

# Meng-kompile kode sumber java menjadi kode byte,
ditaruh di direktori classes
javac -d ../classes HelloGUI.java

# Menjalankan program HelloGui
java -cp ../classes HelloGUI

// Listing-6. HelloGUI.java
// Contoh aplikasi Java memakai Swing GUI

import java.awt.event.*;
import javax.swing.*;

public class HelloGUI {
   public static void main(String[] args) {
  // Utama
  final JFrame frame = new JFrame("Hello");
  JLabel label = new JLabel("Hello World");
  frame.getContentPane().add(label);

  // pernik-pernik Swing
  frame.setSize(200,50);
  frame.setVisible(true);
  frame.addWindowListener(
   new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      frame.dispose();
      System.exit(0);
    }
      });
   }
}
```

Penutup

Bermula dari teknologi coba-coba untuk *embedded* mikro prosesor, siapa kira kini Java menjadi teknologi terpopuler untuk Internet dan sistem enterprise. Kunci suksesnya adalah:

- **Portable**. Anda bisa memrogram Java di *notebook* dengan sistem operasi Windows kemudian menjalankannya di PC dengan sistem operasi Linux tanpa masalah.
- **Lengkap dan kaya**. Anda bisa membuat berbagai arsitektur program dengan Java. Bandingkan jika Anda harus belajar C++, Perl atau Phyton dan PHP, atau ASP untuk hal yang setara!
- **Mudah dipelajari**. Bahasa Java, boleh dikata adalah yang teranggun (*elegant*) untuk pemrograman berorientasi objek.

Hal yang sering dikeluhkan terhadap Java adalah lambat dan makan memori. Hal ini memang tidak bisa dibantah. Namun, kenyataan menunjukkan bahwa Java sukses di pasar *enterprise computing* dan menjadi *standard platform* di dunia akademik. Dengan demikian, bisa disimpulkan bahwa portabilitas kode, fleksibilitas aplikasi, dan konsistensi pemrograman jauh lebih menarik dibanding hambatannya.

Referensi

1. Java History, *http://java.sun.com/features/1998/05/birthday.html*.
2. Blackdow Java-Linux Installation, *ftp://ftp.uk.linux.org/pub/linux/java/JDK-1.4.1/i386/01/INSTALL-j2sdk*.
3. Java Tutorial: Getting Started, *http://java.sun.com/docs/books/tutorial/getStarted/*.

E. M. Budi *(k0c1l@yahoo.com)*