Jetty: Web Server untuk Java

Apache bukan barang asing di Linux, hampir semua distro menyertakannya. Namun, Anda perlu menambah program agar mendukung halaman berisi kode Java (JSP/servlet). Salah satu alternatif *web server,* yang 100% Java adalah Jetty.

Tiga besar layanan Internet saat ini adalah *Word Wide Web* (WWW), *E-mail* dan *Chatting*. Menurut data statistik yang dikumpulkan Netcraft, pada awal tahun 2003 ada lebih dari 35 juta web server. Dari jumlah itu, lebih dari 60% menggunakan Apache, dan nyaris 30% adalah combo Linux + Apache + MySQL + PHP (LAMP). Sementara itu untuk e-mail, 80% pangsa server dulu dikuasai Sendmail, namun kini mulai digantikan Postfix atau Qmail. Tren terbaru juga menunjukkan bahwa e-mail berbasis web makin diminati, di mana Combo yang populer adalah Linux + Apache + {Postfix/Qmail} + IMAP + {IMP/SquirrelMail/Postman}.

Dengan makin banyaknya layanan dan pilihan, memilih combo yang sinergis tidaklah mudah. Pertimbangan utama biasanya jatuh pada kecepatan, kestabilan dan kemananan. Namun hal tak kalah penting adalah kemudahan pemasangan dan pemaduan. Untuk urusan padu-memadu ini, keseragaman latar belakang teknologi akan sangat menguntungkan.

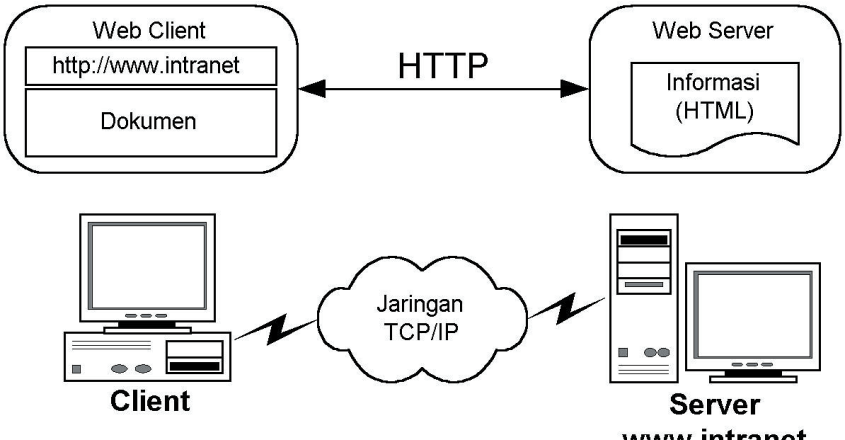
Seri artikel Linux/Java ini memperkenalkan alternatif server Internet berbasis teknologi Java. Akan kita lihat bahwa Java kini sudah cukup matang untuk menyediakan layanan Internet mulai dari WWW, database, chat, maupun e-mail yang dapat dipadukan dengan mulus. Gaya pembahasan adalah belajar melalui contoh. Dengan demikian, Anda dapat langsung mencobanya sendiri, namun untuk konsep yang mendalam Anda harus mencarinya dari sumber lain.

Konsep WWW

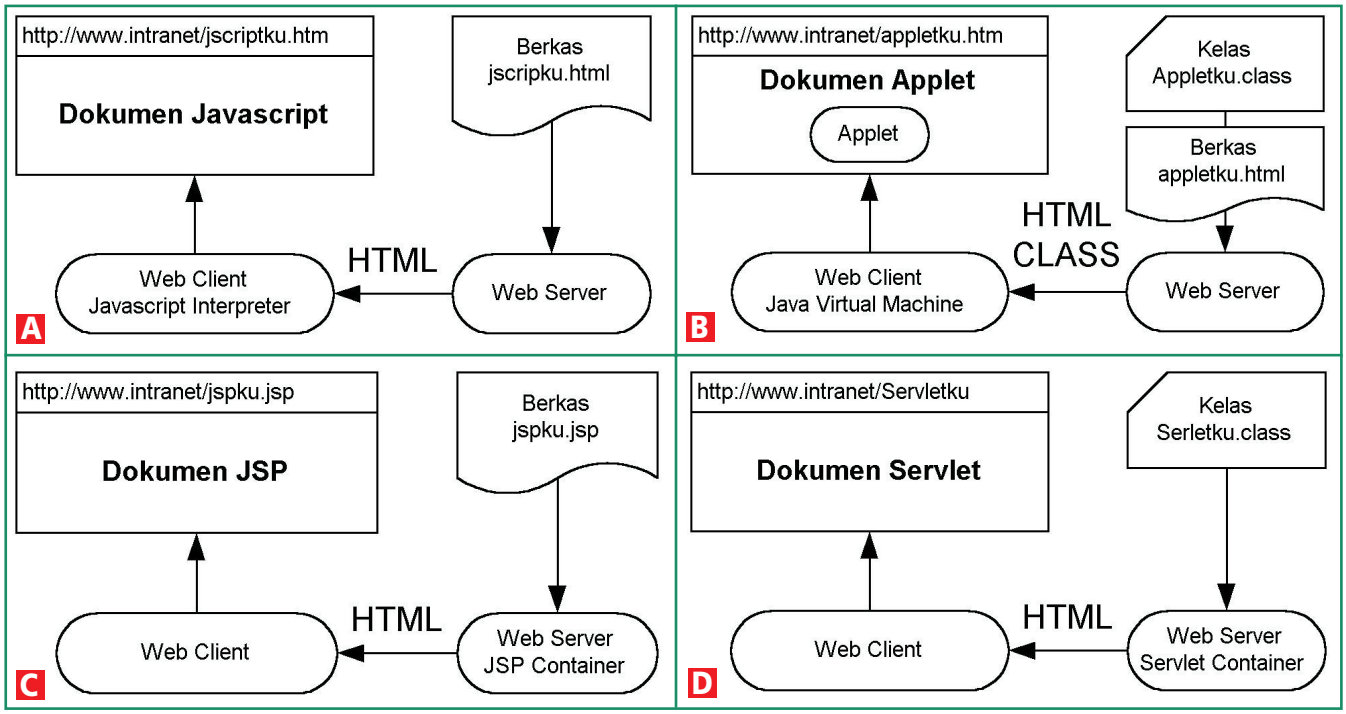
World Wide Web (WWW) merupakan layanan Internet yang memungkinkan manusia memburu informasi dari seluruh dunia. Sistem WWW terdiri atas pasangan *web client* dan *web server* di jaringan TCP/IP (Internet/Intranet). Pemakai yang bekerja di komputer client dapat meminta informasi tertentu dengan mengetikkan alamat *universal resource locator* (URL) web server yang bersangkutan. Di sisi seberang, web server akan menyiapkan informasi yang diminta lalu memberikannya ke web client. Informasi ini kemudian ditampilkan oleh web client sebagai dokumen yang bisa dilihat pemakai di layar.

Tata cara komunikasi antara web client dan web server disebut *Hypertext Transfer Protocol* (HTTP), sedangkan informasi utama yang dikirimkan dari server ke client adalah berkas *Hypertext Markup Langguage* (HTML). Pada dasarnya, HTML adalah informasi teks yang dilengkapi penanda (*tag)* untuk hal-hal berikut:

- Mengatur tata letak teks tersebut seperti jenis huruf, besar huruf, warna, maupun gaya. Tersedia juga penanda untuk tabel, maupun rangka (*frame*).
- Menyelipkan berkas lain yang bisa ditampilkan dalam satu dokumen dengan HTML utama. Ada berbagai bentuk berkas selipan mulai gambar (gif, png, jpg), multimedia (audio, movie), sampai program mini (applet).
- Menyisipkan kode sumber mini (*script*) yang harus dijalankan untuk mendapatkan informasi tertentu sebelum ditampilkan. Ada script yang khusus dijalankan oleh web server dan ada yang dijalankan oleh web client.
- Mengait informasi lain (HTML atau berkas apapun). Kaitan ini ditampilkan ke pemakai dalam bentuk *hyperlink* yang jika diklik oleh pemakai, maka web client akan meminta berkas terkait ke web server untuk ditampilkan atau di-download.
- Mengait program di server. Kaitan ini tampak seperti hyperlink biasa, namun bedanya bila diklik web server akan menjalankan program terkait untuk mendapatkan informasi yang diinginkan. Program yang bisa dijalankan oleh WWW ini disebut *Common Gateway Interface* (CGI).



Gambar 1. Arsitektur dasar WWW



Gambar 2. Cara kerja Javascript (A), Applet (B), JSP (C), dan servlet (D)

Dengan kemampuan yang fleksibel ini, WWW mampu membawa kandungan informasi yang beragam, sekaligus menarik untuk ditampilkan.

Java dan WWW

WWW dan Java bagaikan susu dan kopi. WWW merupakan wahana pembawa informasi, Java membuat informasi tersebut makin enak dinikmati. Ada dua fitur Java yang dapat meningkatkan penampilan dokumen di sisi web client, yakni Javascript dan Applet. Sementara itu dua fitur lain, JSP dan servlet, bekerja di sisi server untuk menghasilkan berkas HTML siap kirim dari data mentah.

Javascript adalah kode sumber mini yang terkandung dalam berkas HTML. Kode sumber ini dikirim apa adanya oleh web server, dan harus dikerjakan oleh web client untuk menghasilkan dokumen. Javascript dikembangkan oleh Netscape dengan meminjam tata penulisan bahasa Java, namun mengalami banyak penyederhanaan dan penyesuaian khusus untuk keperluan WWW. Manfaatnya yang utama adalah untuk mengadaptasi dokumen sesuai lingkungan client dan membuat tampilan dinamis/interaktif sederhana. Javascript biasanya dipadukan dengan *Cascading Style Sheet* (CSS), yakni kode-kode lanjut pengatur gaya tampilan di HTML. Jika Anda ingin membuat halaman web yang ringan dan cepat dimuat, HTML + CSS + Javascript merupakan pilihan terbaik untuk mendapatkan efek yang indah dan dinamis.

Sementara itu, applet adalah program mini yang berjalan di dalam suatu web client. Penanda untuk menyisipkan applet ditulis dalam berkas HTML. Jika menemui penanda ini, web client akan mengambil kode byte applet dari web server, kemudian mengeksekusinya dengan *Java Virtual Machine* (JVM). Applet mampu menampilkan berbagai hal dari sekadar teks bergoyang, animasi grafik, bahkan aplikasi dengan antarmuka grafik lengkap. Sayangnya dari sisi praktis, applet tidak boleh terlalu besar ukurannya, mengingat kecepatan transfer data di Internet sangat terbatas.

Javascript dan applet telah berkembang sejak awal dekade 1990-an. Menjelang akhir milenium kedua lalu, barulah Sun memperkenalkan fitur Java untuk sisi server. Yang pertama adalah servlet, berupa program Java mini di sisi server. Servlet ditulis dengan bahasa Java kemudian dikompilasi menjadi kode byte Java dan dipasang ke web server. Saat client meminta servlet, web server dengan bantuan servlet container akan mengeksekusi kode byte ini untuk menghasilkan informasi (HTML, zip, gambar, dan lain-lain) yang dikembalikan ke web client. Jika dibandingkan, fitur servlet ini mirip dengan *common gateway interface* (CGI) yang biasanya ditulis dalam bahasa Perl atau Phyton. Namun karena berupa kode byte yang dijalankan container, servlet lebih cepat dibanding CGI.

Setelah servlet, kemudian muncul *Java Server Page* (JSP). JSP merupakan halaman HTML yang mengandung kode sumber berbahasa Java (disebut *scriplet*). Saat client meminta halaman JSP, web server akan menugaskan JSP container untuk mengompilasi scriplet dan mengeksekusinya. Keluaran eksekusi ini berupa berkas HTML lengkap yang siap dikirimkan kembali ke client.

Manfaat JSP dan servlet tidak jauh berbeda, yaitu untuk membangun situs WWW otomatis. JSP maupun servlet mampu menghasilkan halaman HTML dari database sehingga cocok untuk

menangani situs-situs seperti portal atau pusat download. Keduanya juga mampu mengambil data dari masukan/keluaran online atau berkomunikasi dengan program lain untuk menghasilkan halaman informasi yang berubah terus secara waktu nyata. Dengan teknologi sisi server ini, web client tidak memerlukan lagi pemrosesan khusus untuk menampilkan dokumen, sehingga cocok untuk konfigurasi *thin client*.

Sampai di sini terlihat bagaimana Java menyediakan pilihan lengkap untuk membangun situs WWW. Dengan satu teknologi, Anda bisa membuat script atau program mini, untuk sisi client maupun sisi server. Tanpa Java, Anda harus menggabungkan beberapa teknologi tak serumpun untuk mendapatkan sistem yang setara.

Web client mampu Java

Untuk menelusuri situs WWW, Anda perlu komputer yang dilengkapi program web client. Pada umumnya web client mampu menjalankan Javascript, namun untuk menjalankan Applet biasanya Anda harus menambahkan Java Run-time Environment (JRE). Ada beberapa pilihan web client mampu Java di Linux. Jika Anda pakai desktop KDE, pilihan otomatis adalah Konqueror (jangan lupa aktifkan Javanya). Jika tidak, Anda bisa pilih Netscape navigator atau Opera. Kali ini kita akan mencoba Opera karena ringan dan cepat. Konfigurasi komputer client minimum adalah sebagai berikut:

- PC Pentium dengan memori 128MB dan ruang harddisk kosong 100MB.
- Linux RedHat 8.0 dengan desktop KDE.
- Network terpasang dengan benar. Pada artikel ini kita akan memakai jaringan Intranet.
- Blackdown Java 2 SDK 1.4.1 telah terpasang (Lihat *InfoLINUX* April 2003)
- Opera 6.12 static-qt dalam paket tar.gz (lihat "Sumber Peranti Lunak").

Untuk memasang Opera, bertindaklah sebagai root di konsole/terminal lalu lakukan langkah-langkah berikut:

Memasang Opera

➔ Uraikan paket Opera ke direktori instalasi sementara

```
mount /dev/cdrom /mnt/cdrom
cd /tmp
tar –xfz /mnt/cdrom/browser/opera-6.12-20030305.1-static-qt.i386.tar.gz
```

➔ Jalankan install.sh

```
cd opera-6.12-20030305.1-static-qt.i386
./instal.sh
```

➔ Hapus direktori install sementara

```
cd ..
rm -Rf opera-6.12-20030305.1-static-qt.i386
```

Install.sh akan menanyakan beberapa pertanyaan, jawab saja semuanya dengan yes. Setelah selesai, Anda dapat menghapus direktori instalasi sementara.

Selanjutnya Anda harus menghidupkan kemampuan Java di Opera. Caranya adalah:

- Jalankan opera
- Pilih menu File – Preference
- Di list pohon sisi kiri, pilih multimedia
- Di sisi kanan akan muncul halaman pengaturan. Aktifkan kotak cek Java enabled.
- Kemudian pada kotak isian Java path, isikan direktori Java plug-in yang sesuai (ada file libjava.so, libawt.so). Untuk Sun atau Blackdown J2 SDK, letak standarnya adalah /usr/java/j2sdk/lib/i386.
- Uji direktori java plug-in, Anda dengan menekan tombol Validate-java-path. Jika masih salah, betulkan isian direktori Anda.
- Setelah selesai, tutup program Opera dan jalankan kembali.

Web server mampu Java

Untuk memproses JSP/servlet, web server memerlukan bantuan servlet container dan JSP container. Program yang paling populer adalah Apache + Tomcat, di mana Apache menjadi HTTP server, dengan Tomcat sebagai servlet/JSP container. Namun saat ini sudah banyak web server yang 100% Java sehingga mampu memproses servlet secara langsung. Produk komersial populer misalnya Caucho Resin, Macromedia JRun, IBM WebSphere, BEA Logic atau Sun Java Embedded Server (lihat *http://directory.google.com/Top/Computers/Programming/Languages/Java/Server-Side/Application_Servers/*). Sementara itu pilihan yang gratis juga banyak, misalnya:

- **Blazix.** Fiturnya banyak walau kecil, sangat cepat, dan mampu membagi beban ke beberapa komputer. Gratis untuk pemakaian nonkomersial.
- **Jetty.** Moduler, kecil dan efisien. Open source.
- **Jo!.** Moduler, mampu membagi beban ke beberapa komputer. Open source.
- **Light Weight Server.** Paling gampang dipasang di Windows. Open source.
- **Bajie.** Cepat dan sangat kecil (kurang dari 1MB) tapi masih beta. Freeware tapi tidak open source.

Kali ini kita akan memakai Jetty yang dibuat oleh Mortbay Consulting, Australia. Jetty memiliki arsitektur moduler sehingga mudah dipakai sebagai komponen program lain, misalnya untuk JBoss atau JOnAS. Kestabilannya tinggi dan sudah cukup matang untuk dipakai di sistem sesungguhnya. Dalam artikel ini, digunakan server sebagai berikut:

- PC Intel Pentium dengan memori 128MB dan ruang harddisk kosong 500MB.
- Linux ClarkConnect 1.2 standard edition (distro khusus server turunan Redhat 7.3, gratis dari *www.clarckconnect.org*).
- Tersambung di network internal dengan IP 10.0.1.1, bernama *www.intranet*.

	Sisi Client	Sisi Server		Sisi Client	Sisi Server	
Script	Javascript	JSP	VS	JScript, VB Script	ASP, PHP	Script
Program Mini	Applet	Servlet		ActiveX	Perl, Phyton, Basic, C	Program Mini

Gambar 3. Teknologi Java untuk WWW dibandingkan pilihan lain

- Blackdown Java 2 SDK 1.4.1 sudah terpasang (Lihat *InfoLINUX* April 2003).
- Jetty web server versi 4.2.9 dalam paket tar.gz.

Pemasangan Jetty, juga aplikasi Java lainnya, sebaiknya dilakukan oleh pemakai khusus di direktori terpisah. Untuk itu, *login*-lah sebagai root, lalu daftarkan group java dan pemakai bernama duke. Buatlah juga direktori /java yang akan menjadi induk direktori seluruh aplikasi java.

Menyiapkan group, user, dan direktori Java

➔ Buat user dan group java

```
useradd -r java
```

➔ Buat user biasa (duke) untuk Java

```
useradd -Gjava duke
passwd duke
```

➔ Buat direktori untuk Java

```
mkdir /java
chgrp java /java
chmod 775 /java
chmod g+xs /java
```

Setelah siap, login-lah sebagai user khusus tersebut (duke) untuk memasang Jetty di direktori /java/server/jetty (disebut JETTY_HOME).

Memasang Jetty

➔ Buat direktori instalasi dan ekstrak paket Jetty

```
mkdir –p /java/server
cd /java/server
tar -xzf /mnt/cdrom/java/server/Jetty-4.2.9.gz.
mv -s jetty.4.2.9 jetty
```

➔ definisikan environment variable JETTY_HOME, sebaiknya tambahkan di login:

```
script /etc/profile.d/setjava.sh
export JETTY_HOME=/java/server/jetty
```

➔ Menjalankan Jetty

```
cd /java/server/jetty
bin/jetty.sh start
```

➔ Menjalan-ulangkan Jetty

```
bin/jetty.sh restart
```

➔ Menghentikan Jetty

```
bin/jetty.sh stop
```

Kini Anda siap menjalankan dan mencoba Jetty. Pergilah ke direktori /java/server/jetty kemudian gunakan script bin/jetty.sh seperti contoh Baris Perintah 3. Tanpa mengubah konfigurasi apa-apa, Jetty akan berjalan di semua IP adress yang ada di komputer pada port 8080. Anda dapat menguji instalasi ini dengan memakai Opera dari komputer client untuk mengakses *http://www.intranet:8080*. Jika semuanya beres, Anda akan melihat halaman percobaan Jetty. Anda juga bisa mencoba demo yang tersedia.

Aplikasi web

Aplikasi web merupakan istilah JSP/servlet untuk isi (*content*) web server yang menyediakan informasi tertentu. Dari pandangan pemakai, aplikasi web diakses dengan URL nama *host* ditambah nama direktori, misalnya. Secara internal, aplikasi web terdiri atas berkas-berkas HTML, JSP, servlet, applet, gambar dan lain-lain yang disusun dalam direktori dengan standar tertentu (**Tabel 1**). Tampak bahwa susunan ini tidak berbeda dengan situs web HTML biasa, kecuali adanya direktori WEB-INF yang menyimpan konfigurasi dan kode byte Java.

Sementara itu, peletakan aplikasi web berbeda pada setiap web server. Pada Jetty ketentuannya adalah sebagai berikut:

- Direktori standard seluruh aplikasi web adalah $JETTY_HOME/webapps (selanjutnya kita sebut WEB_HOME).
- Setiap aplikasi web diletakkan pada direktori $WEB_HOME/<aplikasi>, atau pada berkas $WEB_HOME/<aplikasi>.war. Aplikasi ini diakses pemakai dengan URL http://nama_host/<aplikasi>.
- Aplikasi web utama yang diakses pemakai dengan URL http://nama_host/ (tanpa direktori) harus berupa direktori $WEB_HOME/root atau berkas $WEB_HOME/root.war.

Membuat direktori aplikasi web

Kini kita bisa mulai membuat aplikasi web berisi halaman Javascript, scriplet, servlet maupun applet. Anda dapat melakukannya langsung di web server dengan alat teks editor biasa (misalnya vim). Namun, perhatikan bahwa untuk aplikasi yang besar dan serius, lakukan hal ini di komputer terpisah dengan peralatan HTML/JSP editor khusus, dan setelah jadi barulah dipasang ke server.

Aplikasi yang akan kita buat bernama contoh1. Untuk itu, mula-mula Anda harus membuat direktori $WEB_HOME/contoh1 (selanjutnya kita sebut CONTOH1_HOME) dan beberapa direktori standar lainnya (Baris Perintah 4). Setelah itu, buatlah file WEB-INF/web.xml (Listing 1).

Menyiapkan aplikasi web

➔ Buat direktori aplikasi

```
mkdir -p $JETTY_HOME/webapps/contoh1
export CONTOH1_HOME=$JETTY_HOME/webapps/contoh1
```

➔ Buat direktori-direktori standar

```
cd $CONTOH1_HOME
mkdir -p WEB-INF/classes
mkdir WEB-INF/lib
```

Tabel 1. Direktori Standar Aplikasi Web

Direktori/Berkas	Keterangan
<aplikasi>/	Direktori aplikasi, berisi index.jsp dan berkas-berkas lainnya.
<aplikasi>/<apa_saja>	Direktori untuk mengelompokkan berkas-berkas lainnya (css, template, applet, images, dan lain-lain.)
<aplikasi>/WEB-INF	Direktori tersembunyi untuk menyimpan konfigurasi dan kode byte JSP maupun servlet.
<aplikasi>/WEB-INF/classes	Direktori berisi kode byte Java (*.class).
<aplikasi>/WEB-INF/classes	Direktori berisi pustaka kode byte Java (*.jar).
<aplikasi>/WEB-INF/web.xml	File konfigurasi JSP dan servlet.

➔ Edit file konfigurasi web.xml (Lihat listing 1)

```
vi WEB-INF/web.xml
```

Listing 1. Berkas web.xml paling sederhana

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>

  <display-name>Contoh1</display-name>

  <welcome-file-list>
    <welcome-file> index.jsp </welcome-
    file>
  </welcome-file-list>

</web-app>
```

Membuat halaman JSP

Kini kita bisa mulai membuat halaman web. Mula-mula akan kita buat halaman index.jsp seperti Listing 2. Nampak bahwa halaman JSP tersebut tak ada bedanya dengan halaman HTML biasa, karena belum mengandung scriplet. Coba edit kemudian simpan berkas tersebut sebagai $CONTOH1_HOME/index.jsp. Untuk mencoba web aplikasi ini, jalan ulangkan Jetty. hentikan kemudian mengakses halaman JSP ini dari Opera dengan URL *http://www.intranet:8080/contoh1* (namun sabar, kaitan-kaitannya belum siap).

Listing 2. Berkas index.jsp

```
<HTML><HEAD></HEAD><BODY>
<H1>Aplikasi Web Contoh 1</H1>
* <a href="jspku.jsp">Info JSP</a>
* <a href="jscriptku.jsp">Info
Javascript</a>
* <a href="servlet/servletku">Info
Servlet</a>
* <a href="appletku.jsp">Info Applet</a>
*
</BODY></HTML>
```

Pada halaman JSP, scriplet diselipkan dengan tag <% ... %> . Pada contoh Listing 4, nampak scriplet yang akan memberikan informasi tentang JVM server. Silakan edit listing tersebut sebagai $CONTOH1_HOME/jspku.jsp.

Listing 3. Berkas jspku.jsp

```
<HTML><HEAD></HEAD><BODY>
<h1>Info JSP </h1>

<!-- **** ini scriplet yang menghasilkan
info -->
<%
 out.println("Server : " +
java.net.InetAddress.getLocalHost() + "<br>");
 out.println("Server OS :
" + System.getProperty("os.name") +
   " version
" + System.getProperty("os.version") + "<br>");
 out.println("Server JVM :
" + System.getProperty("java.vm.name") +
    " version " + System.getProperty
("java.vm.version") + "<br>");
%>
<!-- **** -->

<br><a href=index.jsp>Utama</a>
</BODY></HTML>
```

Membuat halaman Javascript

Anda baru saja membuat halaman JSP yang mampu menampilkan informasi tentang server. Jika ingin informasi tentang client, Anda perlu Javascript. Berikut ini contohnya, silakan edit dan simpan sebagai $CONTOH1_HOME/jscriptku.htm.

Listing 4. Berkas jscriptku.htm

```
<HTML><HEAD></HEAD><BODY>
<h1>Info Javascript</h1>

<!-- **** ini Javascript yang menghasilkan
info -->
<SCRIPT LANGGUAGE="JavaScript">
 document.writeln("Client Platform :
" + navigator.cpuClass +
   "/" + navigator.platform + "<br>");
 document.writeln("Client Browser :
" + navigator.appName +
   " version " + navigator.appVersion +
"<br>");
 document.writeln("Java enabled :
" + navigator.javaEnabled() + "<br>");
</SCRIPT>
<!-- **** -->

<br><a href=index.jsp>Utama</a>
</BODY></HTML>
```

Membuat halaman Servlet

Servlet adalah program Java mini yang akan dijalankan server. Tidak seperti scriplet dan Javascript, servlet merupakan kode sumber terpisah yang harus dikompilasi dulu sebelum dipasang ke aplikasi web. Untuk membuat servlet, ada empat hal yang harus dilakukan:

1. Siapkan pustaka java yang diperlukan di direktori $CONTOH1_HOME/WEB-INF/lib. Untuk servlet sederhana, minimal anda perlu javax.servlet.jar.
2. Tulis kode sumber servlet (lihat Listing 5).
3. Kompilasi kode sumber menjadi kode byte di direktori $CONTOH1_HOME/WEB-INF/classes. Setelah dikompilasi kode sumber bisa dipindah karena tidak terpakai lagi oleh aplikasi web.
4. Daftarkan servlet Anda di web.xml (lihat Listing 6).

Mengkompilasi servlet

➔ Salin pustaka java yang diperlukan

```
cd $CONTOH1_HOME/WEB-INF/classes
cp $JETTY_HOME/lib/javax.servlet.jar ../lib
```

➔ Edit kode sumber Java (Listing 5), perhatikan huruf besar!

```
vi Servletku.java
```

➔ Kompilasi kode sumber menjadi kode byte Servletku.class

```
javac -classpath ../lib/javax.servlet.jar -d .
Servletku.java
```

➔ Edit web.xml untuk mendaftarkan servlet (Listing 6)

```
vi ../web.xml
```

Listing 5. Berkas Servletku.java

```
package intranet;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Servletku extends HttpServlet {
 public void doGet(HttpServletRequest
request,
```

```
    HttpServletResponse response)
    throws ServletException, IOException
  {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   /***** Ini perintah servlet yang
membuat berkas HTML */
   out.println("<HTML><HEAD/
><BODY>");
   out.println("<H1>Info Servlet</H1>");

   out.println("Server:
"+java.net.InetAddress.getLocalHost()+"<br>");
   out.println("Server OS :
"+System.getProperty("os.name")+
    " version "+System.getProperty
("os.version")+"<br>");
   out.println("Server JVM :
"+System.getProperty("java.vm.name")+
    " version "+System.getProperty
("java.vm.version")+"<br>");

   out.println("<br><a href=../
index.jsp>Utama</a>");
   out.println("</BODY></HTML>");
    /*****/

  }
}
```

Listing 6. Berkas web.xml dengan pendaftaran Servletku

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/
dtd/web-app_2_3.dtd" >

<web-app>
 <display-name>Contoh1</display-name>

<servlet>
 <servlet-name> Servletku </servlet-name>
 <servlet-class> intranet.Servletku </
servlet-class>
</servlet>

<servlet-mapping>
 <servlet-name> Servletku </servlet-name>
 <url-pattern> servlet/Servletku </url-
pattern>
</servlet-mapping>

<welcome-file-list>
 <welcome-file> index.jsp </welcome-file>
</welcome-file-list>

</web-app>
```

Membuat halaman Applet

Applet, sama seperti Servlet, juga merupakan kode sumber Java yang harus dikompiliasi menjadi kode byte Java. Bedanya, applet akan diambil dan dijalankan oleh web client, sehingga harus diletakkan di direktori yang bisa diakses umum. Pembuatan applet melibatkan 3 langkah utama (Baris perintah 5):

1. Buat kode sumber applet.
2. Kompilasi menjadi kode byte applet di $CONTOH1/.
3. Edit halaman HTML yang menyisipkan applet tersebut.

Membuat applet

➔ Edit kode sumber Java (Listing 7), perhatikan huruf besar!

```
cd $CONTOH1_HOME
vi Appletku.java
```

➔ Kompilasi kode sumber menjadi kode byte (Appletku.class)

```
javac -d . Appletku.java
```

➔ Edit halaman HTML (Listing 8)

```
cd ..
vi appletku.htm
```

Listing 7. Berkas Appletku.java

```
package intranet;

import java.awt.*;
import java.applet.*;
import java.net.*;

public class Appletku extends Applet {
 public void init() {}

 /**** Fungsi applet yang menghasilkan
Info */
 public void paint(Graphics g) {
   try {
    g.drawString("Client : "+
     java.net.InetAddress.getLocalHost(),
10, 50 );
    g.drawString("Client OS :
"+System.getProperty("os.name")+
     " version "+System.getProperty
("os.version"), 10, 75 );
    g.drawString("Client JVM:
"+System.getProperty("java.vm.name")+
     " version "+System.getProperty
("java.vm.version"), 10, 100);
   }
  catch(Exception e) {}
 }
 /*****/

}
```

Listing 8. Berkas appletku.htm

```
<HTML><HEAD/><BODY>
<H1>Info Applet</H1>

<– penanda untuk menyisipkan applet –>
<APPLET code=intranet.Appletku.class
width=500 height=200/>

<br><a href=index.jsp>Utama</a>
</BODY></HTML>
```

Mencoba aplikasi web

Setelah Anda selesai membuat aplikasi web tersebut, jalan-ulangkan Jetty kemudian gunakan Opera di web client untuk melihat *http://www.intranet:8080/contoh1/*. Ikuti kaitan pada index.jsp untuk melihat contoh JSP, Javascrip, servlet maupun applet. Perhatikan bagaimana dengan kode yang mirip, JSP dan servlet memberikan informasi tentang server dan sebaliknya applet memberikan informasi tentang client. Sementara itu, Javascript memiliki kode yang sedikit berbeda untuk memberikan info mengenai web client. Pada Opera cobalah mengubah pilihan *Identify-as* (klik di pojok kanan atas), kemudian muat ulang (*reload*) jscriptku.htm.

Kali ini kita telah memasang web client dan web server yang mendukung Java, dan membuat aplikasi web sederhana. Pada kesempatan berikutnya, kita akan mempelajari konfigurasi lanjut web server seperti virtual host dan Java daemon. Sementara itu, Anda bisa mulai membaca sumber lainnya untuk belajar membuat aplikasi web yang canggih.

E. M. Budi *(k0c1/@yahoo.com)*