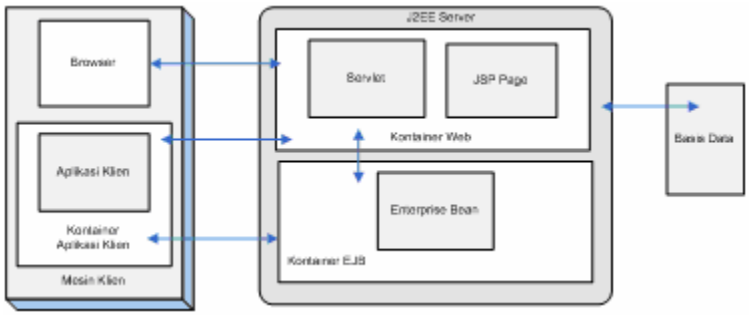
Aturan Praktikum:

- Keterlambatan maksimal 10 menit
- Hasil praktikum, selesai atau tidak selesai ditanggung mahasiswa
- Nilai praktikum adalah nilai perorangan, sebagai tambahan nilai test atau quiz jika nilai test atau quiz kurang mencukupi
- Mahasiswa login ke komputer
- Persiapan dan penjelasan praktikum 10 menit
- Praktikum dilakukan selama 70 menit terdiri dari praktikum sesuai modul
- Hasil praktikum di-zip dan dikumpulkan
- Sisa waktu 20 menit digunakan untuk tes kecil hasil praktikum, saat tes tampilan komputer adalah background desktop dan mahasiswa tidak diperkenankan menggunakan komputer dan melihat modul praktikum
- Mahasiswa logout dari komputer
- Praktikum selesai

SERVLET (BEGINNER)
(Minggu 2 – Praktikum I)

A. Sekilas J2EE

J2EE adalah kumpulan teknologi yang cukup kuat dan berada di atas lingkungan J2SE. J2EE berbasis pada Java2 yang berusaha untuk menyediakan sebuah lingkungan aplikasi yang bersifat *reliable* dan stabil serta dapat dijalankan pada beberapa lingkungan sistem operasi. Teknologi enterprise sebagai perkembangan dari lingkungan Java2 difokuskan pada pemenuhan antarmuka yang standar dimana aplikasi J2EE dapat menghasilkan sebuah aplikasi berbasis server yang tangguh (*robust*) dan tidak bergantung pada lingkungan sistem operasi yang digunakan.



J2EE server menyediakan dua buah kontainer besar yaitu kontainer EJB dan kontainer web dimana kontainer EJB digunakan untuk mengelola dan mengeksekusi Enterprise bean yang juga disebut dengan bean dan kontainer web digunakan untuk mengelola dan mengeksekusi servlet serta *JavaServer Pages* atau yang disebut juga dengan JSP.

Enterprise bean terdiri dari tiga jenis bean yang diantaranya adalah :

- ***Session Bean***
- ***Entity Bean***
- ***Message Driven Bean***

Package pada J2EE dapat berupa Enterprise Archive (EAR), Java Archive (JAR) yang merupakan kumpulan file dalam sebuah paket, dan *Web Archive* (WAR) dimana EAR biasanya merupakan gabungan dari file-file JAR yang biasa digunakan oleh bean, sedangkan WAR biasa digunakan oleh servlet dan JSP.

J2EE memiliki beberapa tipe modul yang diantaranya adalah sebagai berikut :

- EJB, terdiri dari file-file *class* dari enterprise bean dan deskriptor dari EJB *deployment*. EJB biasanya menggunakan *package* berupa JAR dengan ekstensi file .jar.
- Web, seperti servlet dan JSP. Terdiri dari file-file *class* servlet atau file-file *class* yang dibutuhkan JSP, file gambar, file HTML, dan deskriptor dari web *deployment*. *Package* yang digunakan biasanya adalah WAR dengan ekstensi file .war.
- Aplikasi Klien, terdiri dari file class klien dan deskriptor klien. *Package* yang biasanya digunakan adalah JAR dengan ekstensi file .jar.

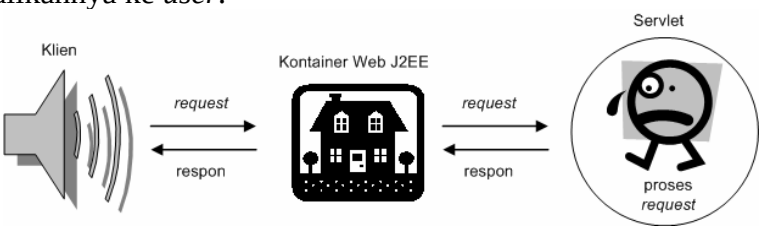
B. Sekilas Servlet

Servlet adalah program Java yang berjalan pada webserver dengan pengaksesan model *request-response*. Daur hidup servlet dikelola oleh kontainer web. Servlet merupakan pondasi pengembangan aplikasi web dengan menggunakan bahasa pemrograman Java. Karena servlet merupakan bagian dari J2EE (*Java 2 Enterprise Edition*) maka servlet dapat menggunakan semua pustaka (*library*) standar Java.

Kelas HttpServlet merupakan kelas turunan dari javax.servlet.Servlet yang melewatkan *request* dan respon melalui protokol HTTP. HttpServlet memiliki metode tambahan yaitu doGet() dan doPost() yang merupakan metode pengiriman *request* melalui protokol HTTP. Servlet adalah *server side scripting* dimana *web server* mengidentifikasi dan menjalankan skrip program yang disisipkan dalam dokumen web sehingga yang dikirim ke klien hanya berupa kode HTML-nya saja.

C. Alur hidup Servlet

Servlet dibangunkan pertama kali oleh kontainer web ketika ada *request* ke servlet kemudian servlet memproses *request* dan memberikannya ke web servlet dan kontainer web mengembalikannya ke *user*.



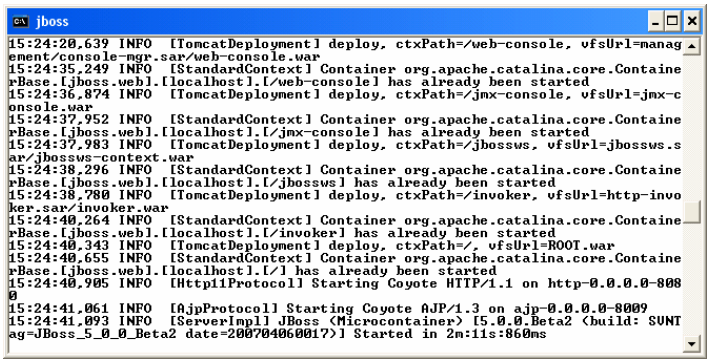
Alur hidup servlet diawali dengan metode init() yang dipanggil oleh kontainer setelah servlet diinisialisasi, kemudian kontainer web memanggil metode service() setelah servlet diinisialisasi, pada tahap ini servlet diijinkan mengakses *request* yang harus diproses oleh servlet dan menghasilkan respon. Kontainer web akan memanggil metode destroy() ketika kontainer web dimatikan atau servlet butuh membebaskan memori.

D. Implementasi J2EE

Sebelum melakukan instalasi terhadap semua perangkat lunak implementasi J2EE, *tool* J2SE harus telah diinstal pada komputer, variabel global JAVA_HOME harus telah diset, dan komputer telah di-*restart* setelah JAVA_HOME diset.

a. JBoss

Salah satu implementasi dari J2EE adalah JBoss. JBoss besifat *open source* dan dapat di-*download* pada http://www.jboss.org. JBoss gratis untuk pengembangan dan *deployment* dan merupakan server aplikasi yang handal dan telah banyak digunakan. Sebelum menjalankan JBoss, pada komputer yang harus sudah terdapat *Java Virtual Machine*, jika belum maka komputer tersebut harus diinstal apa yang disebut dengat *tool* J2SE seperti jdk. Baru setelah itu kopikan *folder* JBoss pada drive C. Untuk menjalankan JBoss, cukup dengan menjalankan C:\<folder JBoss>\bin\run.bat dan untuk menghentikannya tekan ctrl+C. Saat JBoss telah berhasil dijalankan maka akan muncul jendela seperti pada gambar di bawah ini. Dan jika JBoss telah start dengan benar maka akan muncul pesan Started in.



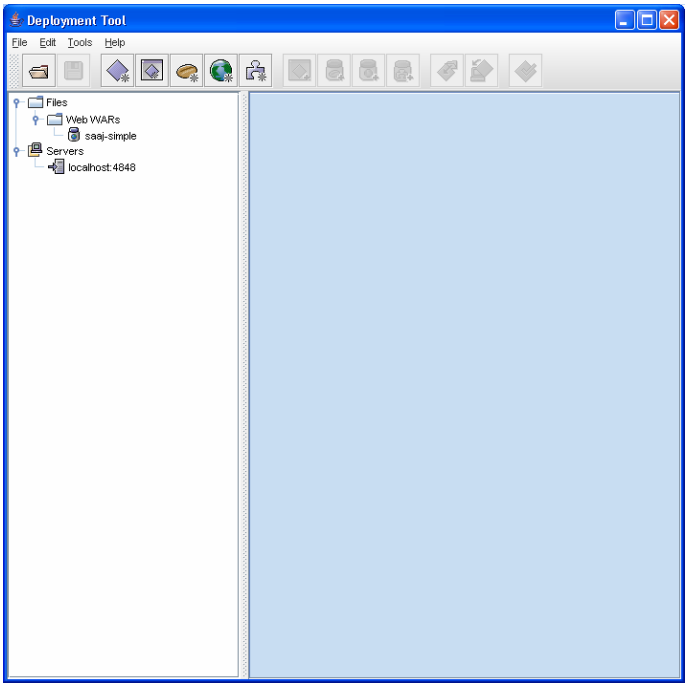
Komponen J2EE yang akan di-*deploy* dengan menggunakan JBoss harus memiliki pemetaan komponen J2EE tersebut untuk dikaitkan dengan JBoss. Biasanya pemetaan ini dimasukkan dalam file jboss-web.xml yang dalam pemaketannya pada servlet atau JSP masuk ke dalam direktori WEB-INF. Servlet atau JSP di deploy pada direktori C:\JBoss\.server\default\deploy\ dengan menyalin file .war ke direktori itu.

b. Server J2EE Sun

Server J2EE Sun atau nama resminya Sun Java System Application Server Platform merupakan implementasi dari J2EE. *Tools* yang ada pada Server J2EE Sun bukan merupakan bagian dari implementasi J2EE, tujuannya lebih pada memberikan kenyamanan pada *programmer*.

Untuk melakukan instalasi Server J2EE Sun cukup dengan menjalankan *file installer*-nya yang bisa di-*download* di http://java.sun.com/j2ee/. Server J2EE Sun dijalankan (*start*) dengan menjalankan Start->All Programs->Sun Microsystems->Application Server PE->Start Default Server. Untuk menghentikan Server J2EE Sun, jalankan Start->All Programs->Sun Microsystems->Application Server PE->Stop Default Server.\

Server Sun memiliki *deploy tool* untuk melakukan *package* terhadap modul servlet maupun JSP menjadi file .war dan men-*deploy* aplikasi ke server. Untuk menjalankannya setelah diinstal klik Start->All Program->Sun Microsystems->Application Server PE->Deploytool maka akan muncul jendela pada gambar berikut.

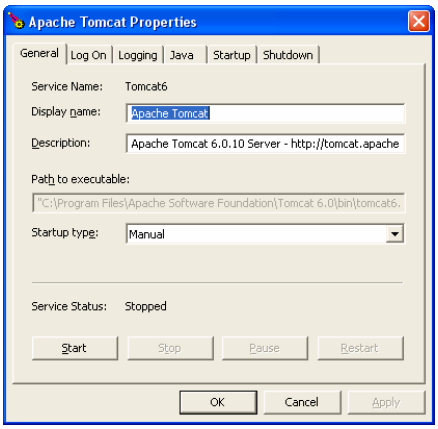


Pada prinsipnya, men-*deploy* aplikasi pada server Sun hanya mengacu path dari file .war, maka jika sebuah file .war di deploy pada server Sun dan dihapus, maka file tersebut tidak dapat diakses lagi.

c. Apache Tomcat

Apache Tomcat merupakan implementasi dari teknologi Servlet dan JSP sehingga hanya dapat digunakan untuk men-*deploy* servlet dan JSP. Apache Tomcat merupakan proyek

open source yang dapat di-*download* di http://jakarta.apache.org/tomcat. Instalasi Apache Tomcat dapat dilakukan dengan mengesekusi *installer* Apache Tomcat. Apache Tomcat dapat dijalankan dengan menjalankan Start->All Programs->Apache Tomcat->Monitor Tomcat hingga muncul jendela seperti pada gambar di bawah ini lalu klik tombol Start, begitu pula jika ingin menghentikan, klik tombol Stop.



Monitor Tomcat biasanya telah dijalankan secara otomatis dan untuk menampilkan jendela seperti pada gambar di atas bisa dengan mengeklik *icon* pada bagian kanan *taskbar* seperti pada gambar berikut:

Pada Tomcat file .war dapat disalin ke folder <INSTAL_TOMCAT>\webapps\ dan me-*restart* Tomcat.

E. Praktikum

a. Persiapan

- Membuat direktori kerja dengan nama SI319-P2-1-Kelas-NIM misalnya SI319-P2-1-A-23507024
- Di dalam direktori di atas, buat direktori Servlet untuk menyimpan file-file yang akan dibuat.

b. Membuat HelloServlet

Susunan file yang harus dibuat di dalam direktori Servlet adalah sebagai berikut:

- Direktori common
  - File build.properties

    - File targets.xml
- Direktori commonweb
    - File build.properties
        - File berisi daftar properti yang akan digunakan build.xml
    - File targets.xml
- Direktori HelloServlet
    - Direktori src
      Berisi file-file *source code*
    - Direktori web
      Berisi file-file html, gambar, dll
    - File build.xml
        - File konfigurasi yang akan digunakan untuk menetukan setting servlet
    - File sun-web.xml
    - File web.xml

File-file pada direktori common dan commonweb digunakan untuk melakukan kompilasi terhadap *source code* java.

Nama file: build.properties (pada direktori common)

```
j2ee.home=C:/Sun/AppServer
j2ee.tutorial.home="C:/Servlet"
sunone.home=${j2ee.home}
admin.password.file=${j2ee.tutorial.home}/common/admin-password.txt
admin.host=localhost
admin.user=admin
admin.port=4848
https.port=8181
domain.resources="domain.resources"
domain.resources.port=8080
db.root=${j2ee.home}/pointbase
db.driver=com.pointbase.jdbc.jdbcUniversalDriver
db.host=localhost
db.port=9092
db.sid=sun-appserv-samples
db.url=jdbc:pointbase:server://${db.host}:${db.port}/${db.sid}
db.user=pbpublic
db.pwd=pbpublic
url.prop=DatabaseName
ds.class=com.pointbase.jdbc.jdbcDataSource
db.jvmargs=-ms16m -mx32m
```

Yang harus diganti dari file di atas jika berbeda path adalah sebagai berikut:

```
j2ee.home=C:/Sun/AppServer   [path   dimana   sun   application   server
diinstal]
j2ee.tutorial.home="C:/Servlet"  [path direktori servlet]
```

Nama file: targets.xml (pada direktori common)

```
  <path id="classpath">
    <fileset dir="${j2ee.home}/lib">
      <include name="j2ee.jar"/>
    </fileset>
  </path>

  <target name="clean" >
    <delete dir="${build}" />
    <delete dir="${dist}" />
    <delete dir="${assemble}" />
  </target>

  <path id="db.classpath">
    <fileset dir="${db.root}/lib">
      <include name="*.jar"/>
    </fileset>
  </path>

  <target name="create-db_common" depends="init"
     description="Create database tables and populate database." >
    <java classname="com.pointbase.tools.toolsCommander" fork="yes" >
      <jvmarg line="${db.jvmargs}" />
      <arg line="${db.driver} ${db.url} ${sql.script} ${db.user}
${db.pwd}" />
      <classpath refid="db.classpath" />
    </java>
  </target>

  <target name="admin_command_common">
    <echo message="Doing admin task ${admin.command}"/>
    <sun-appserv-admin
      command="${admin.command}"
      user="${admin.user}"
      passwordfile="${admin.password.file}"
      host="${admin.host}"
      port="${admin.port}"
      asinstalldir="${j2ee.home}" />
  </target>

  <target name="create-jdbc-resource_common">
    <antcall target="admin_command_common">
      <param name="admin.command"
        value="create-jdbc-resource
         --connectionpoolid ${conpool.name} ${jdbc.resource.name}" />
    </antcall>
  </target>

  <target name="delete-jdbc-resource_common">
    <antcall target="admin_command_common">
      <param name="admin.command"
        value="delete-jdbc-resource ${jdbc.resource.name}" />
    </antcall>
  </target>
```

```
  <target name="deploy-war">
    <antcall target="admin_command_common">
      <param name="admin.command"
        value="deploy ${war.file}" />
    </antcall>
  </target>

  <target name="undeploy-war">
    <antcall target="admin_command_common">
      <param name="admin.command"
        value="undeploy ${example}" />
    </antcall>
  </target>

  <property environment="env" />

  <target name="listprops"
     description="Displays values of some of the properties of this
build file">
     <property file="../../common/admin-password.txt" />

     <echo message="Path information" />
     <echo message="j2ee.home = ${j2ee.home}" />
     <echo message="j2ee.tutorial.home = ${j2ee.tutorial.home}" />
     <echo message="j2ee.home = ${j2ee.home}" />
     <echo message="env.Path = ${env.Path}" />
     <echo message="env.PATH = ${env.PATH}" />
     <echo message="" />
     <echo message="Classpath information" />
     <echo message="classpath = ${env.CLASSPATH}" />
     <echo message="" />
     <echo message="Admin information" />
     <echo message="admin.password = ${AS_ADMIN_PASSWORD}" />
     <echo message="admin.password.file = ${admin.password.file}" />
     <echo message="admin.host = ${admin.host}" />
     <echo message="admin.user = ${admin.user}" />
     <echo message="admin.port = ${admin.port}" />
     <echo message="https.port = ${https.port}" />
     <echo message="" />
     <echo message="Domain information" />
     <echo message="domain.resources = ${domain.resources}" />
     <echo message="domain.resources.port = ${domain.resources.port}"
/>
     <echo message="" />
     <echo message="Database information" />
     <echo message="db.root = ${db.root}" />
     <echo message="db.driver = ${db.driver}" />
     <echo message="db.host = ${db.host}" />
     <echo message="db.port = ${db.port}" />
     <echo message="db.sid = ${db.sid}" />
     <echo message="db.url = ${db.url}" />
     <echo message="db.user = ${db.user}" />
     <echo message="db.pwd = ${db.pwd}" />
     <echo message="url.prop = ${url.prop}" />
     <echo message="ds.class = ${ds.class}" />
     <echo message="db.jvmargs = ${db.jvmargs}" />
```

```
  </target>
```

Nama file: build.properties (pada direktori commonweb)

```
#war
build=build
sql.script=books.sql
war.file=${example}.war
assemble=assemble
assemble.war=${assemble}/war
```

Nama file: targets.xml (pada direktori commonweb)

```
<target name="prepare" depends="init"
  description="Create build directories.">
    <mkdir dir="${build}" />
  </target>


  <target name="copy" depends="prepare"
     description="Copy HTML and JSP pages" >
    <copy todir="${build}">
      <fileset dir="web">
        <include name="**/*.html" />
        <include name="**/*.jsp" />
        <include name="**/*.jspf" />
        <include name="**/*.jspx" />
        <include name="**/*.gif" />
        <include name="**/*.xml" />
        <include name="**/*.tld" />
        <include name="**/*.tag" />
        <include name="**/*.jpg" />
        <include name="**/*.css" />
      </fileset>
    </copy>
  </target>

  <target name="create-war" depends="build"
       description="Packages the WAR file">
     <echo message="Creating the WAR...."/>
     <delete file="${assemble.war}/${war.file}" />
     <delete dir="${assemble.war}/WEB-INF" />
     <copy todir="${assemble.war}/WEB-INF">
       <fileset dir=".">
         <include name="*.xml" />
         <exclude name="build.xml" />
         <exclude name="web.xml" />
       </fileset>
     </copy>
     <copy todir="${assemble.war}/WEB-INF/classes/">
       <fileset dir="${build}">
         <include name="**/*.class" />
       </fileset>
     </copy>
```

```
    <copy todir="${assemble.war}/WEB-INF/tags">
      <fileset dir="${build}">
        <include name="*.tag" />
      </fileset>
    </copy>
    <copy todir="${assemble.war}/WEB-INF">
      <fileset dir="${build}">
        <include name="*.tld" />
      </fileset>
    </copy>
    <copy todir="${assemble.war}">
      <fileset dir="${build}">
        <include name="*.jsp" />
        <include name="*.gif" />
      </fileset>
    </copy>
    <war destfile="${assemble.war}/${war.file}"
         webxml="./web.xml" filesonly="true" >
      <fileset dir="${assemble.war}" includes="WEB-INF/**, *.jsp,
*.gif" />
    </war>
    <copy file="${assemble.war}/${war.file}" todir="." />
  </target>

  <target name="copy-clock" depends="build" if="clock.exists"
       description="Copies clock class for bookstore2">
    <copy file="${build}/clock/DigitalClock.class"
          todir="${assemble.war}/clock/" />
  </target>

  <target name="listprops-web" depends="init,listprops"
      description="List property values">
    <echo message="sql.script = ${sql.script}"/>
    <echo message="conpool.name = ${conpool.name}"/>
    <echo message="jdbc.resource.name = ${jdbc.resource.name}"/>
  </target>
```

Nama file: build.xml (dalam direktori HelloServlet)

```
<!DOCTYPE project [
  <!ENTITY targets SYSTEM "../common/targets.xml">
  <!ENTITY webtargets SYSTEM "../commonweb/targets.xml">
]>

<project name="hello" default="build" basedir=".">
  <target name="init">
      <tstamp/>
  </target>

  <!-- Configure the context path for this application -->
  <property name="example" value="hello" />

  <!-- Configure properties -->
  <property file="../common/build.properties"/>
  <property file="../commonweb/build.properties"/>
```

```
  &targets;
  &webtargets;

  <target name="build" depends="copy"
     description="Compile app Java files" >
    <javac srcdir="src" destdir="${build}">
      <include name="**/*.java" />
      <classpath refid="classpath"/>
    </javac>
  </target>

</project>
```

Nama file: web.xml (dalam direktori HelloServlet)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" version="2.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>Hello</display-name>
</web-app>
```

Nama file: sun-web.xml (dalam direktori HelloServlet)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
Application Server 8.0 Servlet 2.4//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_2_4-0.dtd">

<sun-web-app>
  <context-root>/Hello</context-root>
</sun-web-app>
```

Nama file: HelloServlet.java (dalam direktori HelloServlet/src/)

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.naming.*;
import java.io.*;
import java.rmi.RemoteException;
import java.util.logging.*;
import javax.rmi.PortableRemoteObject;


public class HelloServlet extends HttpServlet {

    private void writePage(HttpServletRequest request,
     PrintWriter out) {
      out.println("<html>"
         + "<head><title>Hello</title></head>");
      out.println("<body>");
      out.println("<h3>Hello World!</h3>");
      out.println("<br><hr><br>");
```

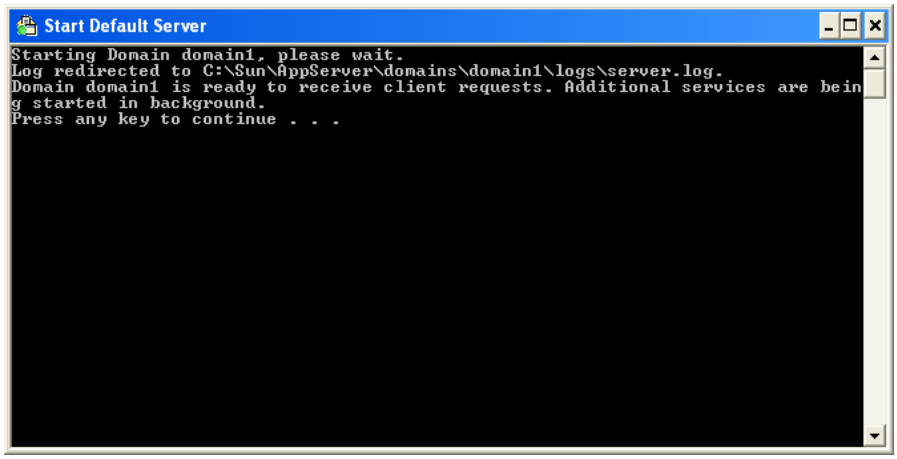
```
      out.println("</body></html>");
    }

    public void doGet(HttpServletRequest request,
     HttpServletResponse response)
     throws ServletException, IOException {
      response.setContentType("text/html");
      response.setBufferSize(8192);

      PrintWriter out = response.getWriter();
      writePage(request, out);
      out.close();
    }

    public void doPost(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
      response.setContentType("text/html");
      response.setBufferSize(8192);

      PrintWriter out = response.getWriter();
      writePage(request, out);
      out.close();
    }

    public String getServletInfo() {
      return "Hello servlet mengatakan hello world.";
    }

}
```

Method yang harus ada pada servlet walaupun tidak berisi apapun adalah:

- doGet(HttpServletRequest request, HttpServletResponse response)
  Method yang dijalankan untuk variabel get
- doPost(HttpServletRequest request, HttpServletResponse response)
  Method yang dijalankan untuk variabel post

c. Menjalankan Servlet

1. Buka Start->All Programs->Sun Microsystems->Application Server PE->Start Default Server sampai keluar command prompt

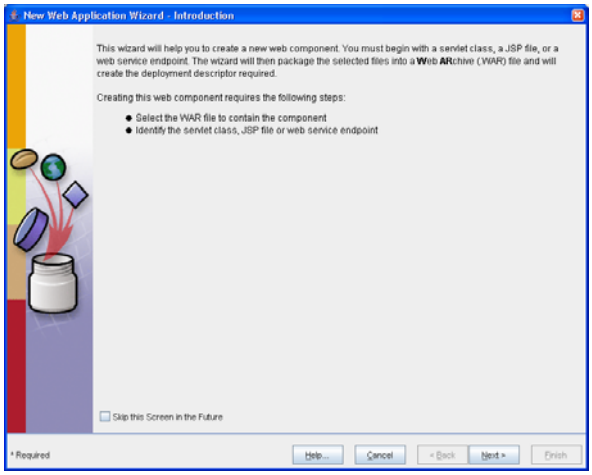


Tekan enter.

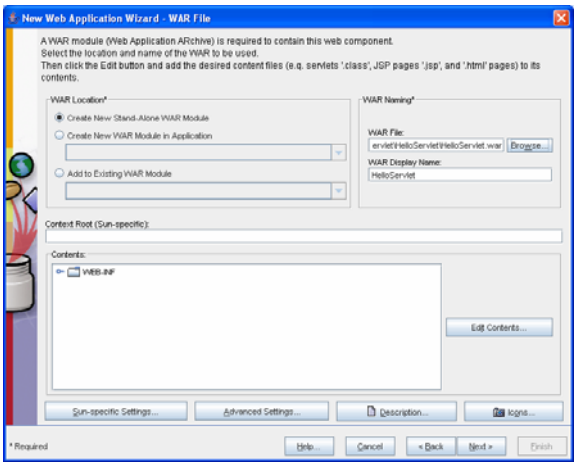
2. Buka jendela command prompt, masuk ke direktori Servlet
3. ketik perintah $>asant build. Perintah asant build digunakan untuk mengkompilasi *source code*, jika *source code* telah lulus kompilasi maka akan muncul pesan sebagai berikut:

```
Buildfile: build.xml

init:

prepare:

copy:

build:

BUILD SUCCESSFUL
Total time: 2 seconds
```

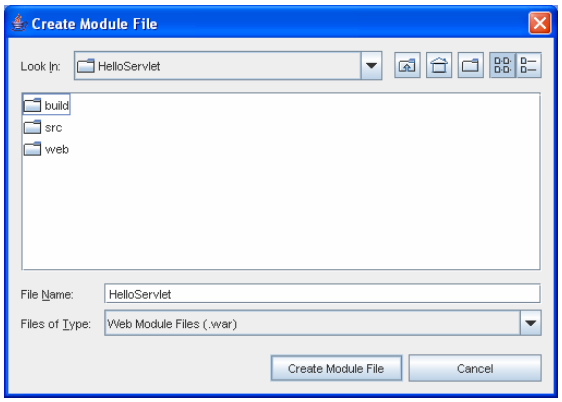
4. Buka Deploytool dari Sun untuk melakukan *packaging* menjadi file WAR. Klik Start->All Program->Sun Microsystems->Application Server PE->Deploytool.
5. Klik File->New->Web Component... . Hingga muncul jendela seperti pada gambar berikut.



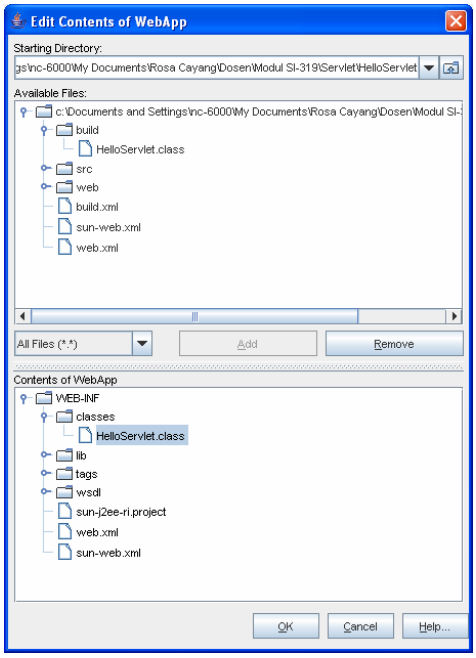
6. Klik Next. Hingga muncul jendela seperti pada gambar berikut.



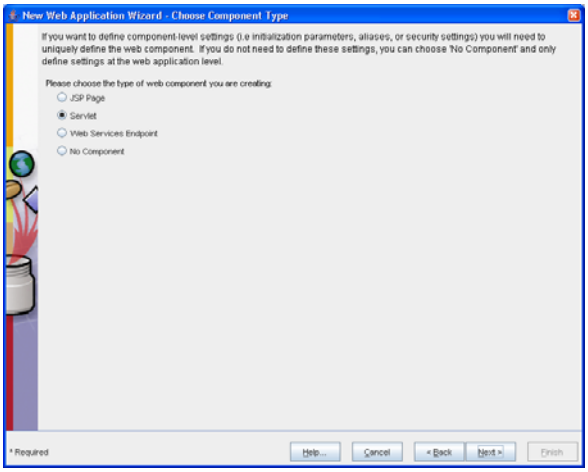
Pilih Create New Stand-Alone WAR Module, isi *field* WAR file dengan mengeklik browse untuk mencari direktori HelloServlet, dan isi file name dengan HelloServlet seperti pada gambar berikut dan klik create module file.



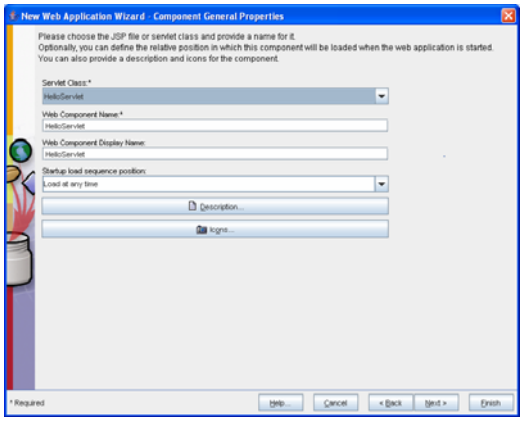
Klik edit contents (gambar sebelumnya) hingga muncul jendela seperti gambar berikut. Tambahkan direktori contoh pada available file dengan mengeklik direktori build, kemudian klik file HelloServlet.class dan klik tombol add kemudian klik ok. Kemudian klik next.



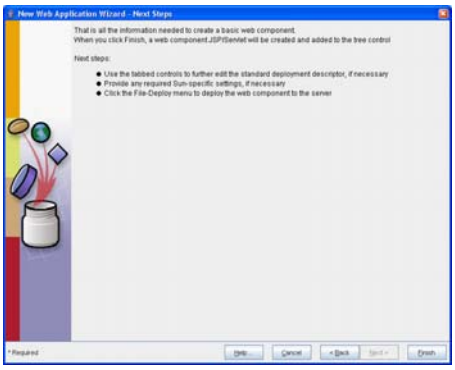
7. Pada jendela seperti gambar berikut, pilih servlet. Klik next.



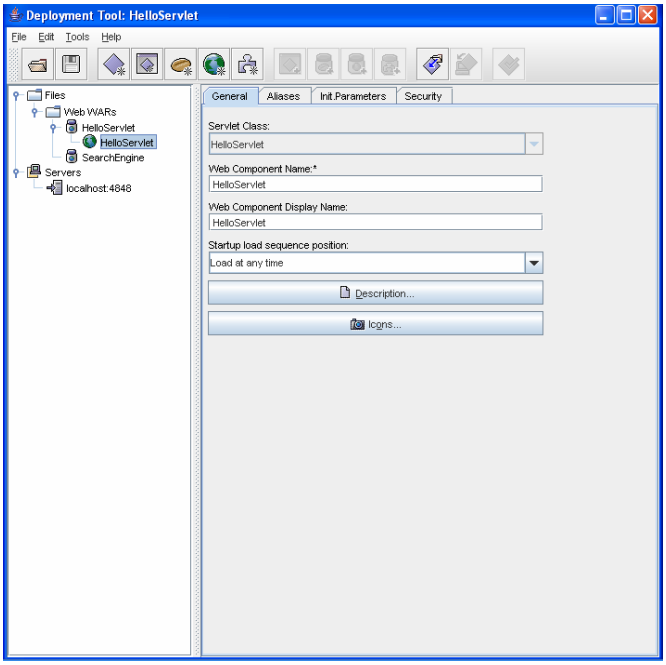
8. Kemudian akan muncul jendela seperti gambar berikut. Isi field yang ada dengan mengeklik tanda panah dan pilih pilihan yang ada, maka beberapa *field* akan terisi. Klik next.



9. Setelah muncul jendela seperti pada gambar berikut, klik Finish.

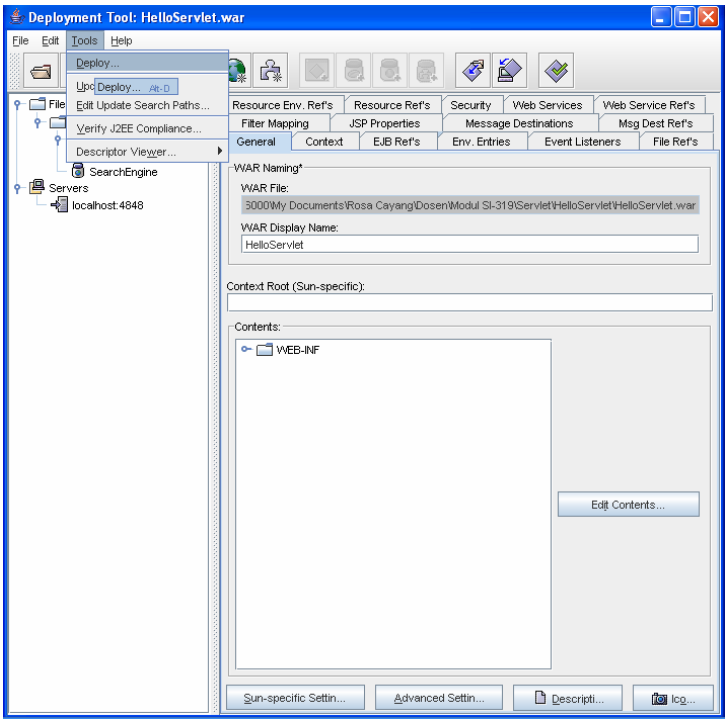


10. Setelah kembali ke jendela awal, pilih tab Aliases, klik Add dan isi *field* Aliases dengan / lalu tekan enter.

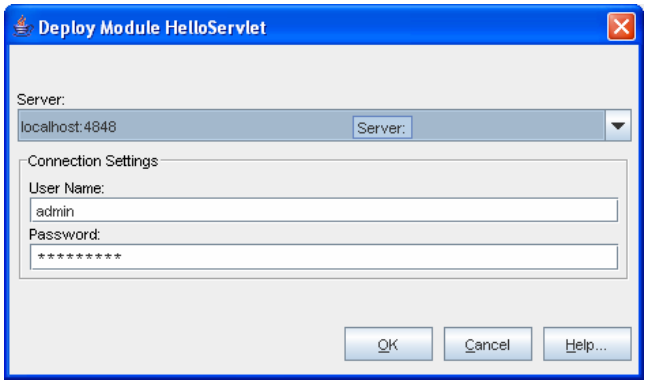


11. Setelah kembali ke jendela awal, klik tombol save yang bergambar disket, maka file WAR akan terbentuk di dalam direktori HelloServlet.

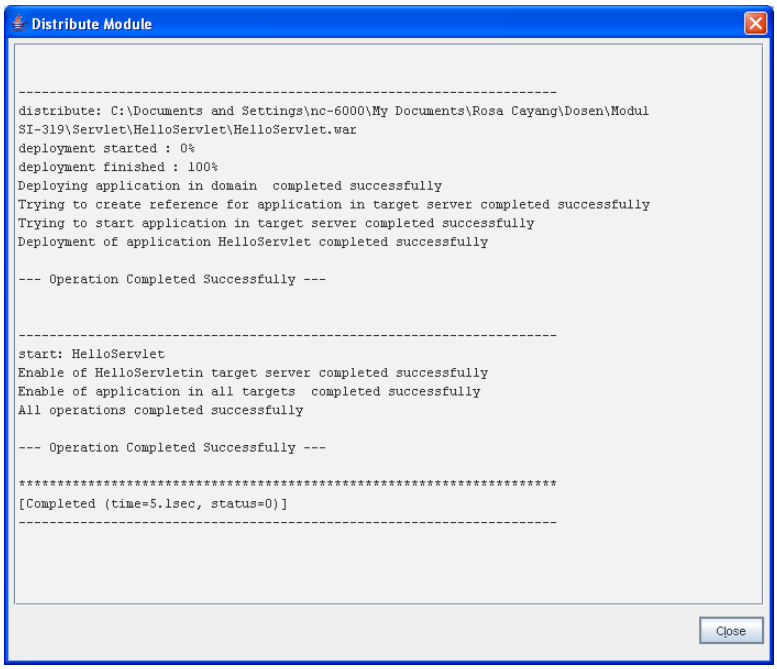
12. Pilih Tools->Deploy seperti pada gambar berikut:



13. Hingga muncul jendela berikut, lalu klik ok



14. Hingga muncul jendela berikut, klik close:



ketikkan http://localhost:8080/HelloServlet/

jika tampilan bukan “Hello World!” maka cek port mungkin salah dengan membuka file C:/Sun/AppServer/domains/domain1/config/domain.xml

cek pada bagian file

```
<http-service>
        <http-listener acceptor-threads="5" address="0.0.0.0" blocking-
enabled="false"      default-virtual-server="server"      enabled="true"
family="inet" id="http-listener-1" port="1309" security-enabled="false"
server-name="" xpowered-by="true">
    </http-listener>
```

Untuk contoh bagian file di atas berarti port yang digunakan bukan 8080 tapi 1309