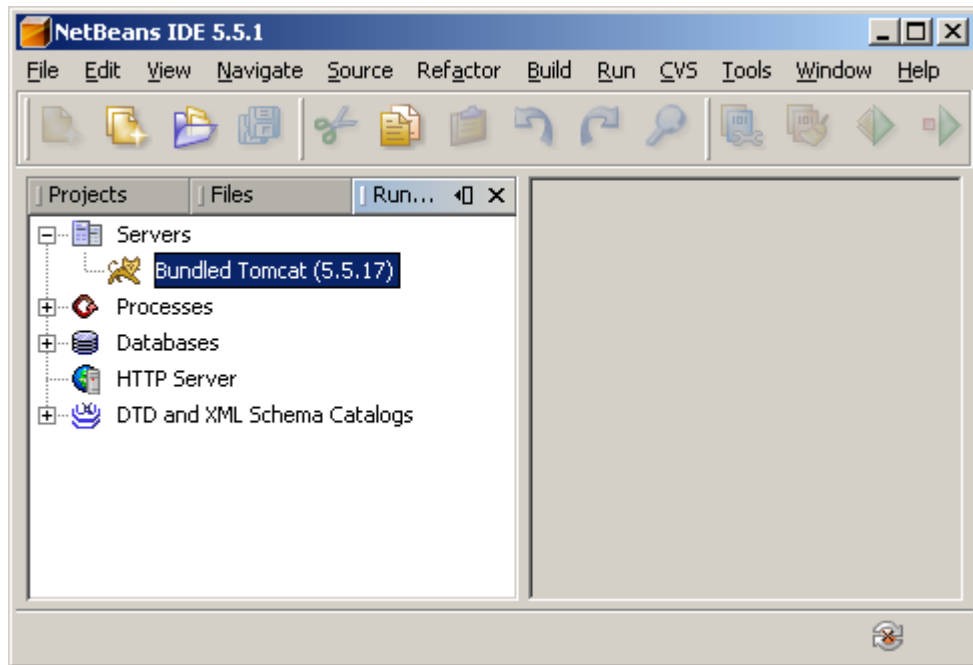


Tutorial JSP dengan IDE Netbeans

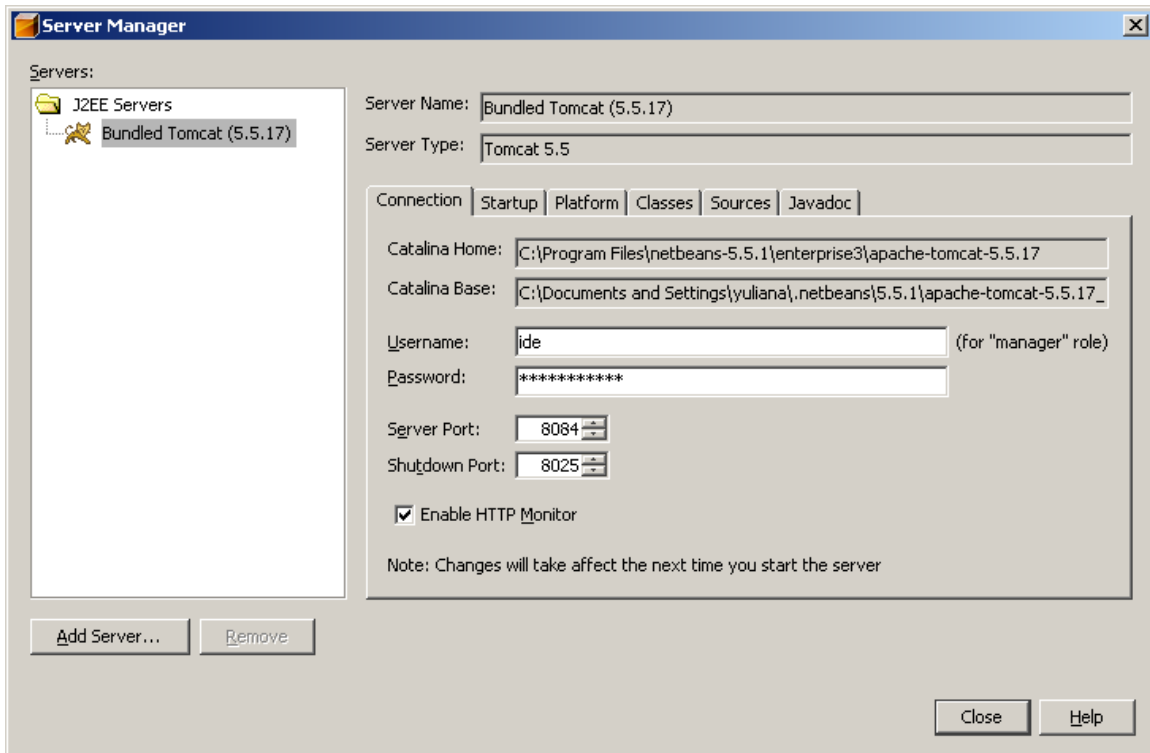
Server Tomcat

Ketika kita melakukan pemrograman web dalam IDE Netbeans, ada dua keuntungan bila memilih Tomcat sebagai server Java. Yang pertama, ia bebas biaya lisensi. Yang kedua, IDE Netbeans telah menyertakan bundled-Tomcat di paket instalasinya.

Setelah menjalankan Netbeans, buka jendela Runtime lalu buka node Servers. Didalamnya telah ada Bundled-Tomcat 5.5.x dimana x tergantung dari versi Netbeans yang digunakan.



Untuk melakukan kustomisasi terhadap server Tomcat ini, klik kanan node Bundled-Tomcat lalu buka menu Properties (atau dari IDE dengan menjalankan menu Tools | Server Manager)



Ketika melakukan instalasi Netbeans, IDE akan membuat nama user default = “ide” dan password = “default”. Password default ini dapat dilihat dalam file bernama “tomcat-users.xml”. Ada dua file tomcat-user.xml. Satu terletak dalam folder instalasi Netbeans (CATALINA_HOME) dan satu lagi terletak dalam user-directory (CATALINA_BASE). Untuk melihat lokasi kedua folder ini, lihatlah dialog Server Manager seperti ditunjukkan dalam gambar di atas. Carilah file tomcat-users.xml dalam folder CATALINA_Base\conf. Buka dengan editor teks lalu lihat password untuk user bernama ide.

Start, Stop dan Administrasi

Untuk menjalankan fungsi start (menjalankan server), stop(menghentikan server) dan administrasi terhadap server Bundled Tomcat ini, lakukan klik kanan di atas node Bundled Tomcat dalam jendela Runtime. Untuk melakukan fungsi administrasi, jalankan menu View Admin Console.

Jika Anda menjalankan menu View Admin Console maka akan terbuka web-browser default. Browser akan membuka halaman dengan URL <http://localhost:8084/admin/> Didalamnya Anda harus mengisikan user dan password. Klik tombol login sehingga Anda masuk ke halaman Administration Tool.

Sekilas Tentang JSP (Java Server Pages)

JSP merupakan perluasan dari teknologi servlet. Tujuan dari JSP adalah untuk lebih menyederhanakan penulisan servlet. JSP sebelum dijalankan oleh server, akan dikompilasi terlebih dahulu menjadi servlet, meskipun proses ini tidak terlihat oleh kita.

JSP dan servlet dapat dipakai bersama-sama dalam sebuah aplikasi web.

Perbedaan utama antara servlet dan JSP adalah, untuk servlet layer aplikasi tidak sepenuhnya terpisah dari layer presentasi, dimana logika aplikasi atau logika bisnis berada di dalam file program Java. Sedangkan presentasi diletakkan dalam output berupa content yang dihasilkan juga oleh servlet.

JSP sendiri lebih menitikberatkan pada aspek presentasi ketimbang aspek aplikasi. Untuk JSP, kode Java dan HTML digabungkan dalam satu file, yaitu file dengan ekstensi *.jsp. Dalam JSP, layer presentasi boleh dikatakan terpisah dari logika aplikasi atau logika bisnis.

Bahkan dalam perkembangannya sekarang JSP dapat saja tidak mengandung kode Java sama sekali. Beberapa logika pemrograman Java dapat digantikan oleh tag library. Misalnya JSTL (*Java Server Page Standar Tag Libray*) dapat mengenali beberapa logika pemrograman seperti loop dan kondisional.

JSP dan ASP

Perbedaan yang pasti, standart JSP dikeluarkan oleh Sun Microsystems, sedangkan ASP dikeluarkan oleh Microsoft. JSP memakai komponen JavaBeans yang tidak tergantung platform (*platform-independence*) sehingga JSP bisa dikenali oleh aplikasi yang berjalan dalam platform apapun. Sementara itu ASP memakai obyek COM (*Component Object Model*) yang spesifik untuk platform Microsoft (Windows) seperti misalnya ADO.

Dalam ASP, script dapat ditulis memakai VBScript atau Jscript. Dalam JSP, script ditulis dengan memakai bahasa Java, serta dapat memakai berbagai framework dan *standart tag library*.

Dari sisi layanan server, ketika browser mengirimkan request dengan ekstensi *.asp, maka server web akan menyerahkan obyek request kepada interpreter ASP yang kemudian akan mengambil dan menjalankan skrip. Sedangkan untuk JSP, ketika browser mengirimkan request dengan ekstensi *.jsp, maka server web akan menjalankan script sebagai servlet, dimana halaman JSP akan dikompilasi terlebih dahulu menjadi servlet. Setiap instance dari servlet akan berjalan dalam thread tersendiri.

JSP dan Servlet

JSP merupakan perluasan dari servlet dan memiliki beberapa keunggulan. Yang pertama adalah bahwa kode yang ditulis untuk JSP relatif lebih ringkas. Yang kedua, proses deployment lebih mudah. Sebuah file JSP dapat diperlakukan sama seperti file HTML ketika dilakukan deployment. Dari segi pemakaian komponen JavaBeans, JSP relatif lebih mudah melakukannya.

JSP menggabungkan kode Java dan content (teks statik, kode HTML, XML, kode lainnya seperti skrip dan tag). Ini membuat content tidak perlu seluruhnya dihasilkan oleh program Java. Misalnya kita dapat langsung memakai tag-tag HTML dalam file JSP.

Servlet seluruh content harus dihasilkan oleh program Java di dalam servlet sehingga kita tidak bisa memakai tag-tag HTML secara langsung tanpa kode Java. Namun dari sisi developer hal ini juga mengharuskan programmer Java dan programmer web (content) bekerja sama dengan baik sebab kedua macam kode (Java dan content) akan digabungkan dalam sebuah file.

Untuk servlet, developer dituntut untuk berfungsi baik sebagai programmer Java dan programmer web. Sedangkan untuk JSP, programmer Java dapat bekerja secara terpisah dari programmer web meskipun tetap diperlukan sinkronisasi antar keduanya. JSP akan dikompilasi menjadi servlet ketika dijalankan. Kode servlet dari JSP tidak terlihat oleh programmer ketika dilakukan debugging. Ini bisa menyulitkan, terutama untuk mencari lokasi kesalahan ketika terjadi run-time error.

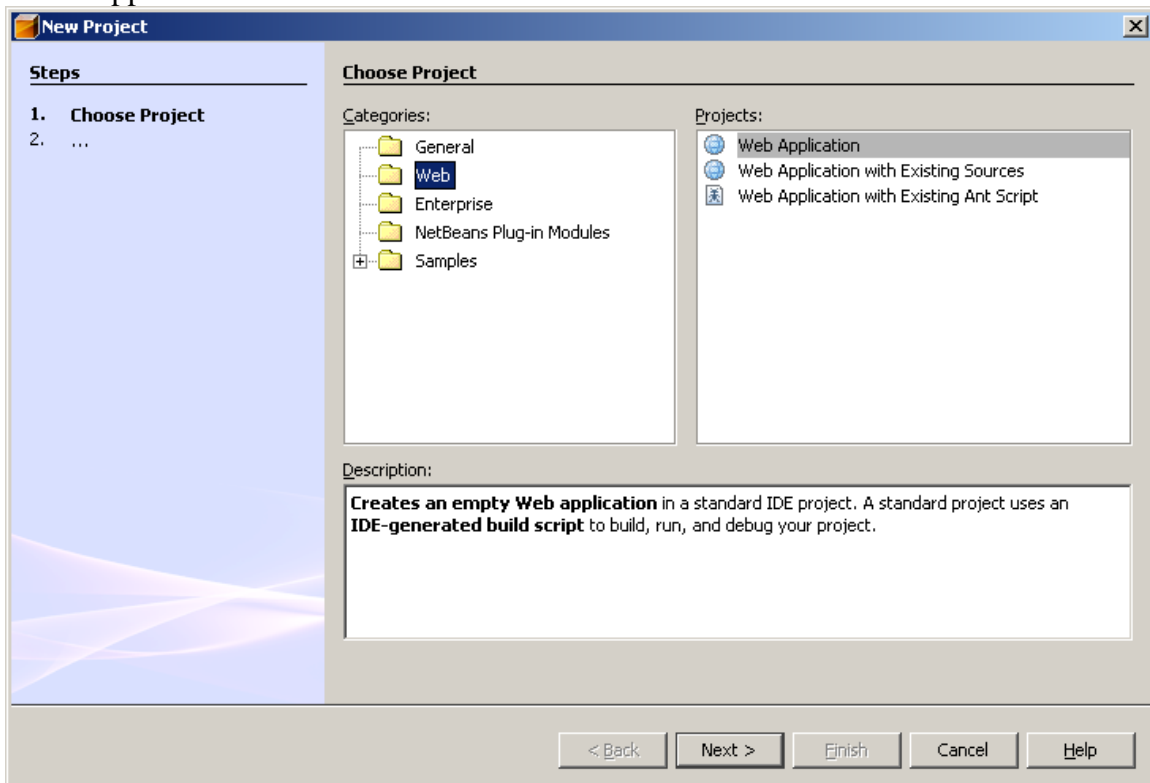
Server

Beberapa server pendukung JSP adalah :

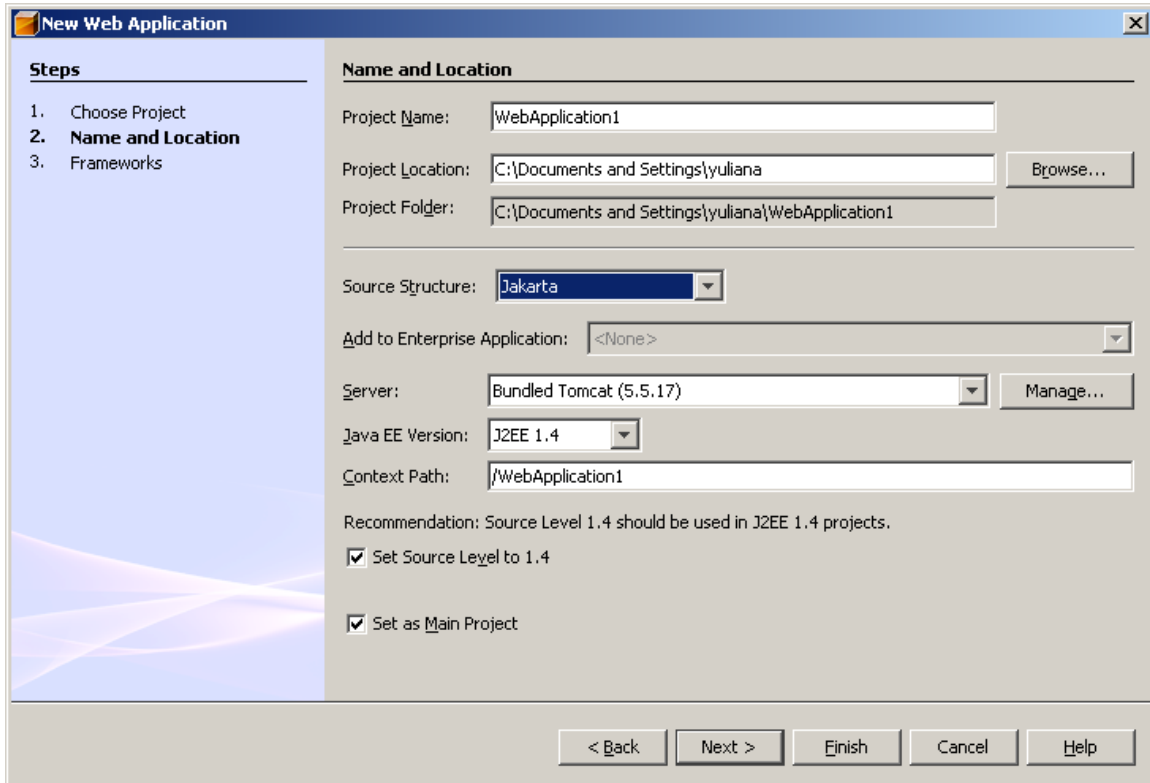
- Apache Jakarta Tomcat(<http://jakarta.apache.org/tomcat/>)
- Sun Java System Application Server
- Sun Java System Web Server
- GlassFish
- JBoss
- Oracle Application Server

Aplikasi Web dalam Netbeans

Buat sebuah project baru pilih File | New Project pilih Categories “Web” dan project “Web Application”



Langkah kedua dari wizard aplikasi web ini, masukkan data yang berkaitan dengan identitas project serta data yang berkaitan dengan server dan versi J2EE.

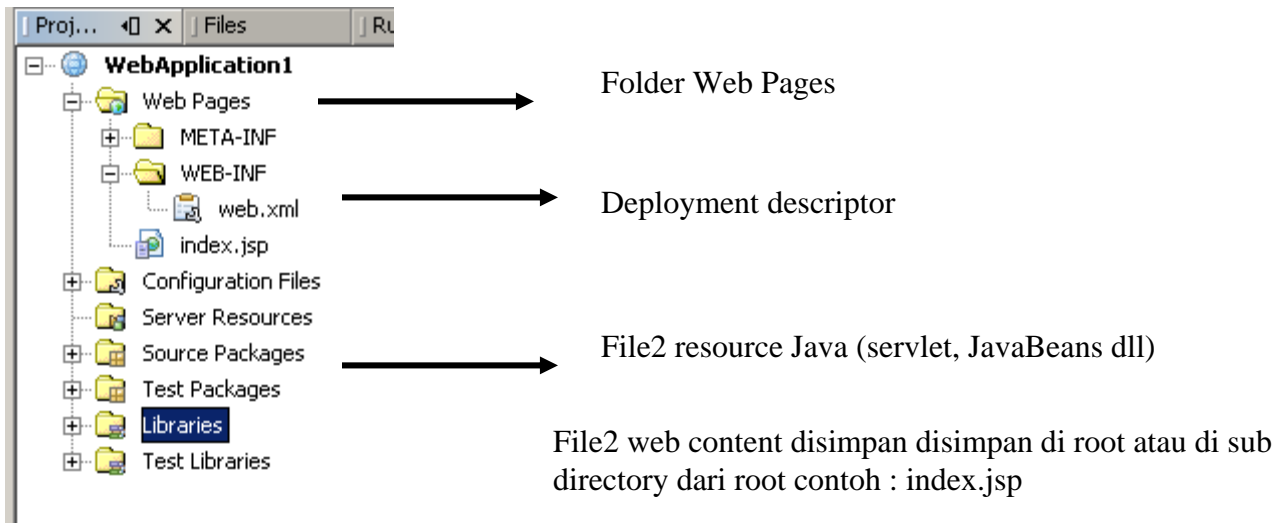


Setelah membuat project aplikasi web maka kita dapat menambahkan ke dalam project file-file *web resource* berupa obyek class Java (seperti servlet, JavaBean dan file Java lainnya), file-file *web-content* (HTML, JSP, image dan sebagainya), library dan sebagainya. Untuk menambahkan library, lakukan klik kanan pada node Libraries dalam jendela project dan jalankan menu kontekstual Add Project, Add Library atau Add JAR/Folder.

Web Pages dan File WAR

Ketika kita membuat aplikasi web dalam IDE Netbeans maka semua file yang akan dipakai sebagai web-content harus diletakkan di dalam folder bernama Web Pages. Anda dapat meletakkannya dalam root-directory dari folder tersebut atau bisa juga dalam sub-directory di dalamnya.

File-file web-content adalah file-file yang dapat diakses oleh browser dalam mesin client. Folder Web Pages akan terlihat dalam jendela Projects sebagai node.



Terlihat dalam node Web Pages terdapat node META-INF dan WEB-INF. Kedua node folder ini merupakan folder yang khas di dalam sebuah aplikasi web.

Folder META-INF dan WEB-INF berisikan *deployment descriptor* yang berupa informasi yang diperlukan oleh web server ketika aplikasi di deploy. Informasi ini disimpan dalam bentuk file-file *deployment descriptor* berupa file XML.

File web.xml adalah file wajib yang harus ada dalam sebuah aplikasi web. File ini akan dipakai oleh setiap web-server yang mendukung teknologi JavaServer. Informasi dalam file web.xml akan menunjukkan bagaimana melakukan deployment terhadap komponen-komponen yang ada dalam aplikasi web.

File context.xml terdapat dalam folder META-INF. File ini khusus dipakai untuk server Tomcat. Untuk aplikasi web sederhana, umumnya kita tidak perlu mengedit file context.xml ini karena telah berisi informasi yang cukup bagi aplikasi web. File context.xml biasanya berisikan context-path dari aplikasi web serta konfigurasi tingkat lanjut dari aplikasi web. Misalnya elemen-elemen dalam file ini dapat berisikan informasi mengenai file log dari servlet, JNDI resource, parameter untuk JDBC, context-parameter dan sebagainya.

Setelah dilakukan build (compile) terhadap proyek aplikasi web ini maka file-file web-content serta file-file *.class akan disimpan dalam file WAR. File WAR adalah file Web-archieve (identik dengan JAR, Java archieve). File hasil kompilasi ini dapat Anda lihat dalam node “dist” dalam jendela Files. File WAR adalah file yang akan di deploy dalam server Java. Struktur dari file WAR dalam jendela Files diatas mengikuti struktur yang ditentukan oleh proyek Jakarta. Secara garis besar struktur ini dapat dijabarkan sebagai berikut :

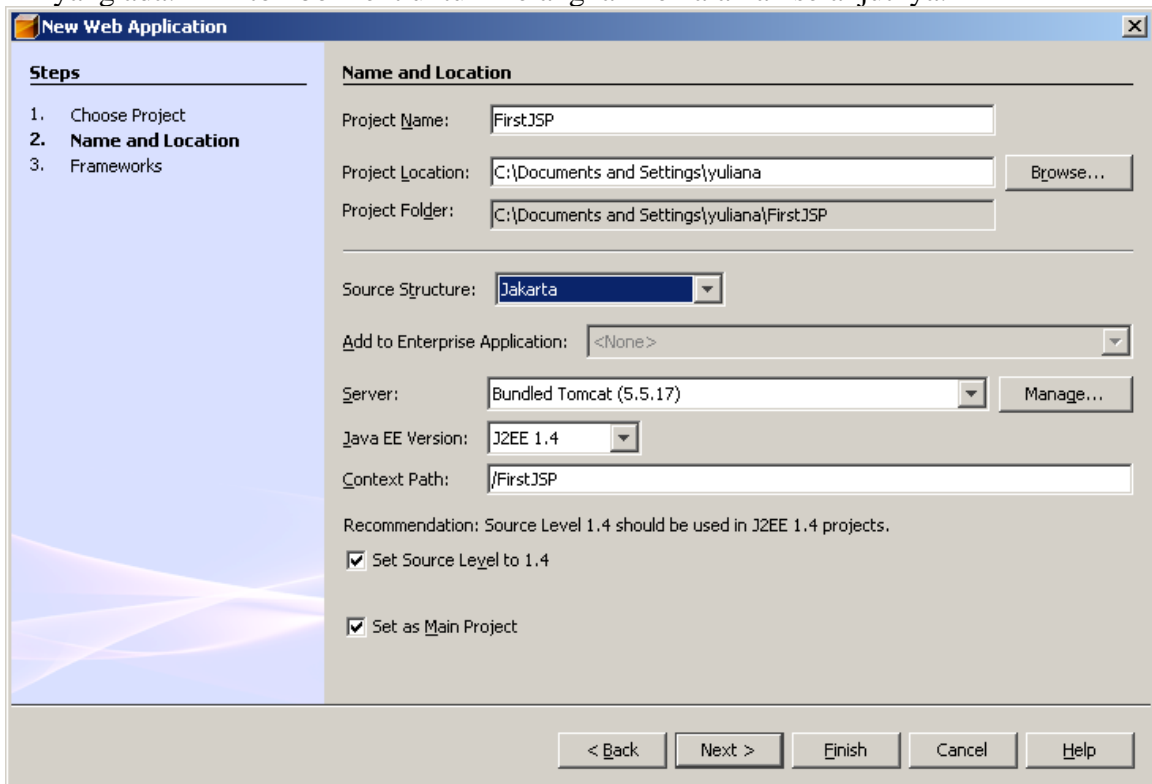
- Folder root yang berisikan semua file dan sub folder aplikasi. Folder ini juga dikenal sebagai document-base
- Sub-direktori dan file-file web-content dalam folder root. File-file ini dapat diakses langsung oleh browser client, misalnya file HTML, image dan file JSP.

- Folder WEB-INF, yang berisikan file deployment-descriptor (web.xml). Folder ini juga dapat berisikan sub-direktori classes, lib dan tags, sub-direktori lain dan file-file lainnya. Folder ini dan isinya tidak dapat diakses langsung oleh client.
- Folder classes (dalam WEB-INF) yang berisikan file-file *.class hasil build dari file-file Java, misalnya file-file servlet, JavaBeans dan file-file pendukung lainnya.

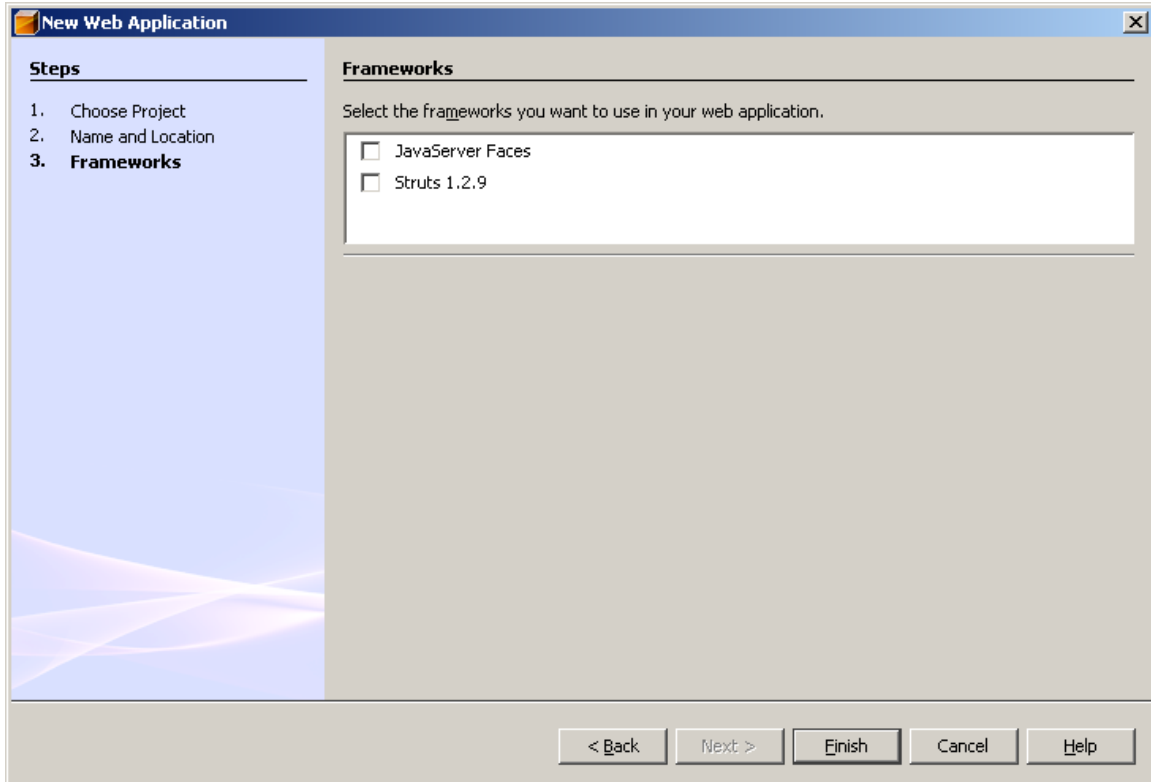
Contoh JSP Sederhana

Lakukan langkah-langkah seperti berikut:

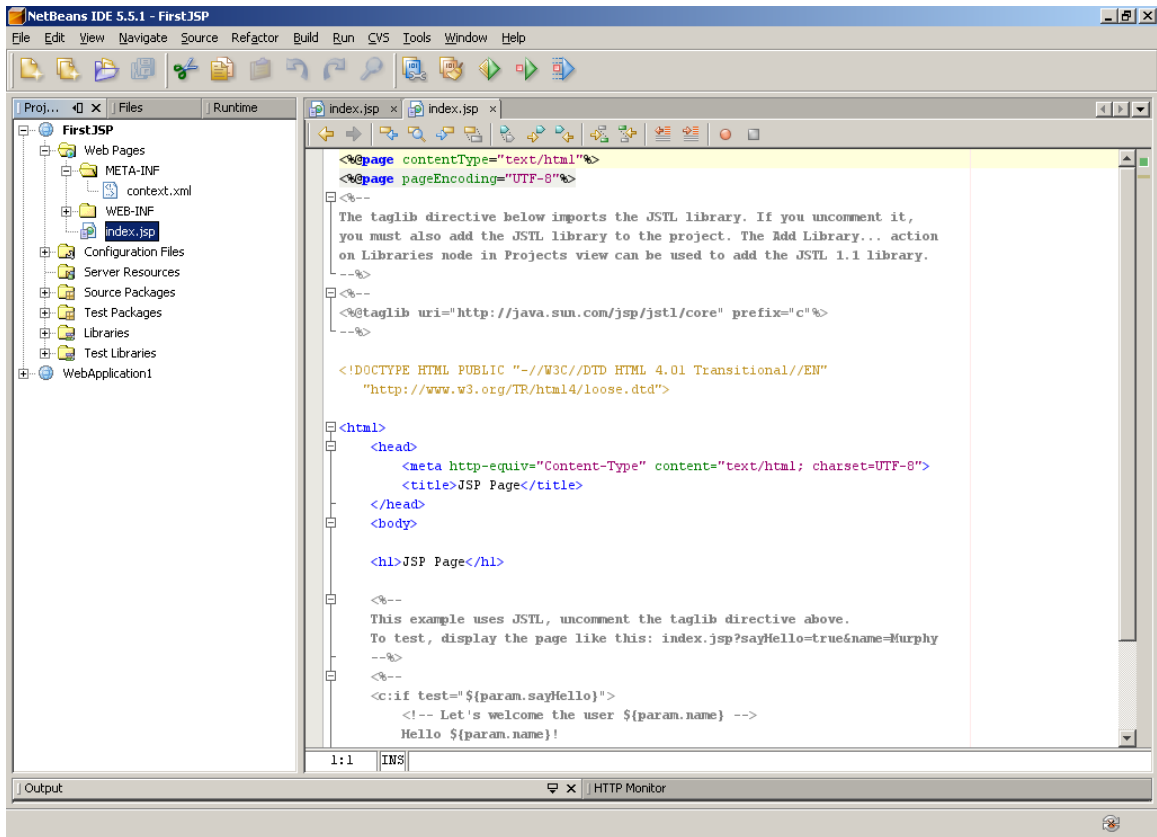
1. jalankan menu File | New Project
2. Pilih Categories “Web” dan File Types = “Web Application”. Setelah itu klik tombol next.
3. Isi Project Name = “FirstJSP”. Pilih Project Location, pilih source structure = “Jakarta”, karena kita akan memakai server Jakarta Tomcat. Pilih server = “Bundled Tomcat”. Pilih J2EE Version = “J2EE 1.4” (ini tidak berarti kita harus memiliki instalasi J2EE). Isi Context Path = “/FirstJSP”. Centang semua checkbox yang ada. Klik tombol next untuk melangkah ke halaman selanjutnya.



4. Dalam halaman selanjutnya Anda dapat memilih framework. Dalam contoh kali ini kita tidak memakai framework. Oleh karena itu klik finish.



Dalam jendela Projects terdapat struktur aplikasi web, sedangkan dalam source editor akan dibuka file “index.jsp”



Ubahlah file index.jsp seperti berikut:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
</head>
<body>

<h1>JSP Page</h1>

<%
out.println("Selamat Datang <br> ");
out.println("Selamat Datang <br> ");
%>

</body>
</html>
```

Build dan Deploy

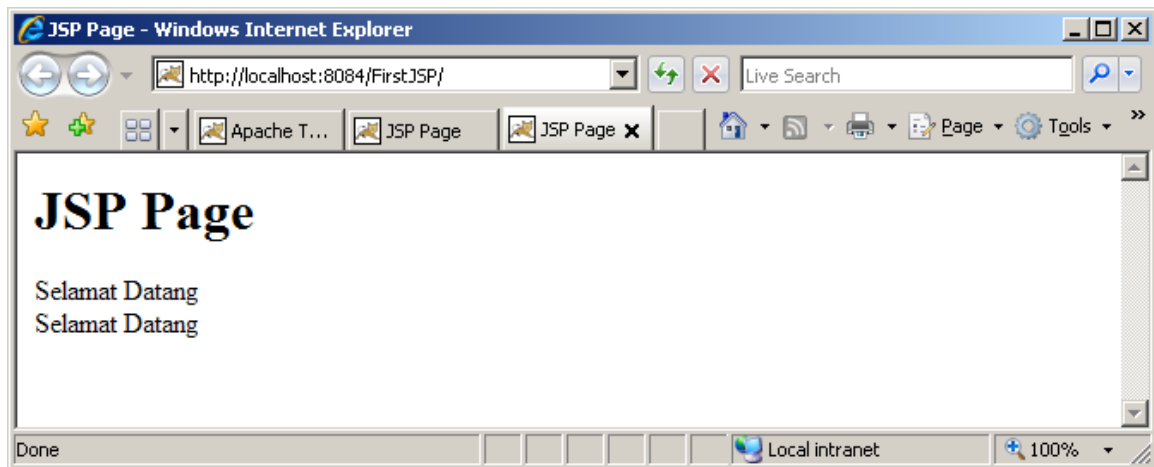
Proyek harus dibangun terlebih dahulu :

- Dalam jendela Projects, klik kanan node project FirstJSP lalu jalankan menu Build Project. Jika sukses maka akan ditampilkan pesan sukses di dalam jendela Output. Setelah itu lakukan deployment ke dalam server Tomcat.
- Dalam jendela Projects, klik kanan node project FirstJSP lalu jalankan menu Deploy Project

Menjalankan Project

Setelah proses build dan deployment maka project dapat dijalankan:

- Dalam jendela Projects, klik kanan node project FirstJSP lalu jalankan menu Run Project. Web-browser default akan dibuka.



Context Path

Ketika membuat project aplikasi web, kita akan diminta untuk menentukan context path. Path ini digunakan untuk membedakan resource dari satu aplikasi web dengan resource aplikasi web lain yang ada dalam satu server.

Context-path adalah bagian dari request-URI (Uniform Resource Identifier). Request-URI sendiri terdiri dari context-path, servlet-path dan path-info. Konteks dari aplikasi web akan mempengaruhi URL dari web-content. Bentuk umum dari namespace URL untuk mengakses content dari aplikasi web (melalui protokol HTTP) adalah sebagai berikut:

http://hostname:port/context/servlet_atau_jsp

Contoh :

Hostname = "localhost"

Port = "8080"

Context-path = "/MyServlet"

Komponen web = class servlet bernama "HelloServlet"

Nama package dari class servlet = "myservlet"

Maka client akan mengakses servlet “HelloServlet” memakai format berikut:
`http://localhost:8080/MyServlet/myservlet.HelloServlet`

Dimana `context-path="/MyServlet"`, `servlet-path="/myservlet.HelloServlet"` dan `path-info=null`.

Contoh:

```
Hostname = "www.nawolo.com"  
Port = default port HTTP  
Context-path = "/MyJSP"  
Komponen web = File JSP bernama "hello.jsp"
```

Maka client akan mengakses servlet “HelloServlet” memakai format berikut:

```
http://www.nawolo.com /MyJSP/hello.jsp
```

Dimana `context-path="/MyJSP"`, `servlet-path="/hello.jsp"` dan `path-info=null`.

Contoh sebuah request-URI yang memiliki path-info:

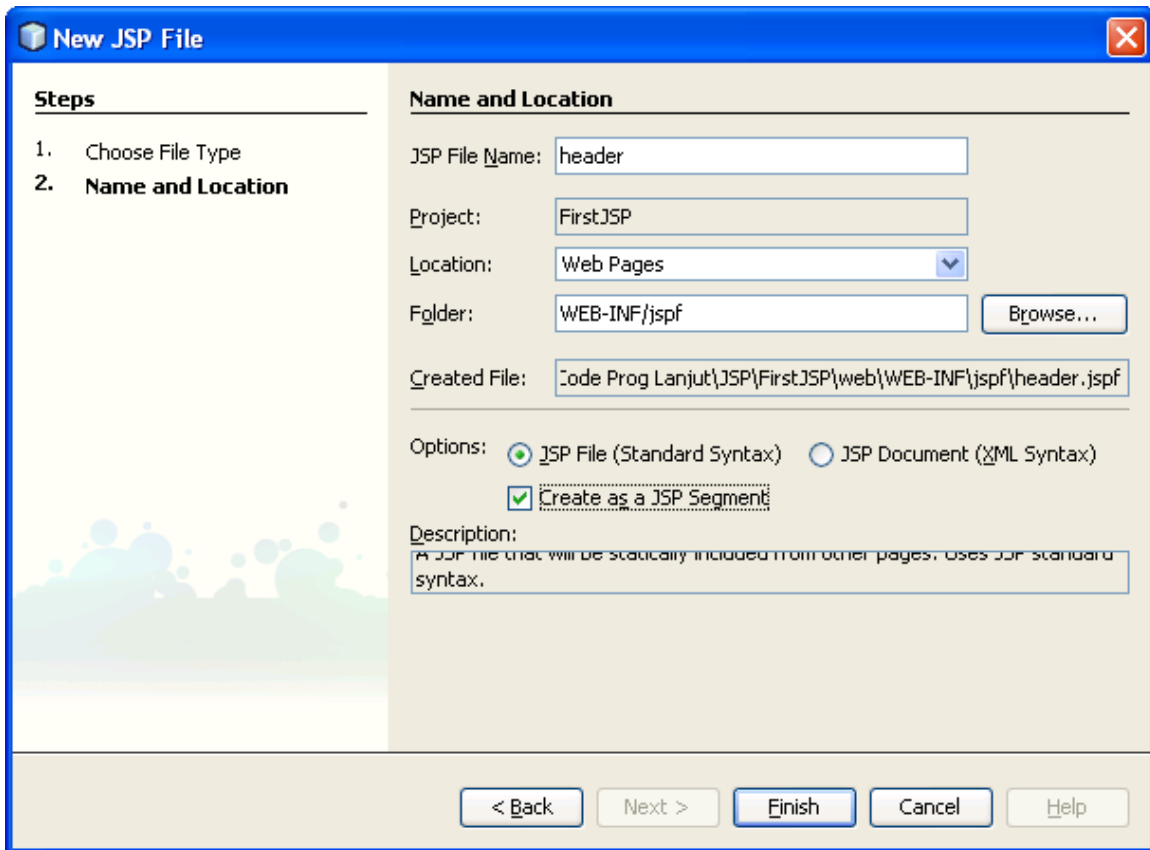
```
http://localhost:8080/MyServlet/HelloServlet/index.html  
path info = "/index.html"  
http://localhost:8080/MyServlet/HelloServlet/help/  
path info = "/help"
```

Dalam IDE Netbeans kita dapat menentukan context-path di dalam wizard ketika membuat aplikasi web. Atau bisa juga diubah di dalam project dialog Project Properties.

File Include/Segmen

Kita akan menambahkan file JSP yang akan diincludekan ke file JSP lainnya. File ini disebut juga sebagai file segmen, dengan ekstensi *.jspx dan diletakkan dalam folder “/WEB-INF/jspf”

1. Dalam jendela Projects, bukalah folder WebPages dan kemudian klik kanan di node WEB-INF. Buat folder jspx dan pada folder tersebut jalankan menu kontekstual New | JSP
2. Dalam wizard, isi JSP File Name=”header”. Pilih location Web Pages. Isi folder = “WEB-INF/jspx”. Pilih option = “JSP File (Standart Syntax)”. Centang check-box “Create as a JSP Segment”. Selanjutnya klik tombol finish.



Hasilnya dalam jendela Projects, file header.jspf akan ditambahkan dalam node Web Pages | WEB-INF | jspf. File juga akan terbuka dalam source editor. Isi file header.jspf

header.jspf

```

<%@ page import = "java.util.*, java.text.*" %>

<%
    DateFormat df = DateFormat.getDateInstance(DateFormat.LONG) ;
    String sd = df.format(new Date()) ;
%>

<h4><%=sd%></h4>

```

File ini akan menampilkan tanggal hari ini dengan format lokal. Format diambil dengan class DateFormat, dengan style LONG. Hasil format yang berupa string kemudian disimpan dalam variabel sd. Output tanggal dinyatakan dengan tag ekspresi <%=sd%>. Format HTML <h4> akan membuat font menjadi lebih besar dan tebal.

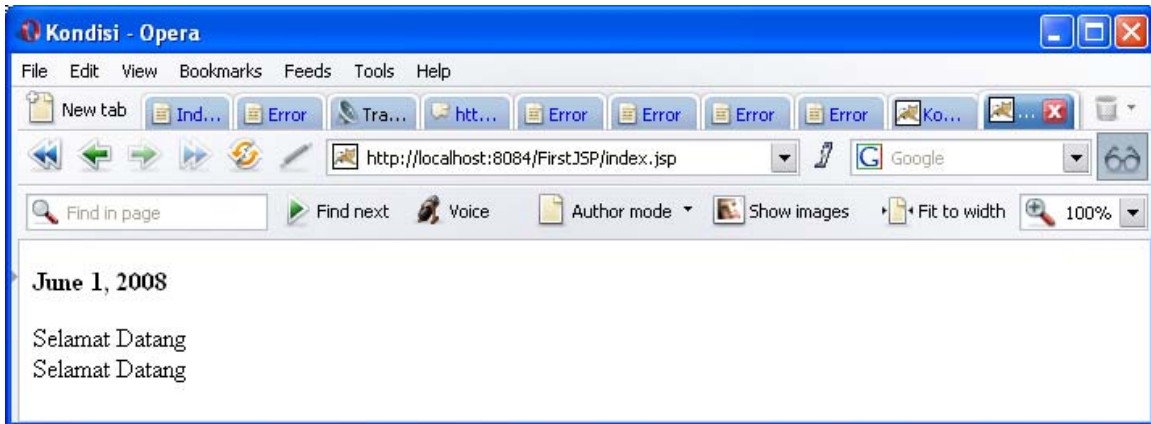
File ini akan diincludekan ke file JSP lain menggunakan include directive:

```

<%@ include file = "/WEB-INF/jspf/header.jspf" %>

```

Sebagai uji coba includekan kedalam file index.jsp yang telah kalian buat. Hasilnya adalah sebagai berikut:



JSP dan Applet

Kekurangan JSP dan servlet adalah tidak memiliki kemampuan untuk menampilkan komponen-komponen AWT dan Swing. Kelemahan ini dapat diatasi dengan memakai applet atau Japplet dari Swing.

Untuk memakai applet pada JSP, kita dapat memakai tag HTML :

```
<APPLET CODE="MyApplet.class" WIDTH=400 HEIGHT=300>
</APPLET>
```

Cara lain adalah memakai elemen action `<jsp:plugin>`. Elemen ini dapat dipakai dengan cara yang identik dengan tag `<APPLET>`. Hanya saja penulisan atribut harus dalam huruf kecil.

```
<jsp:plugin type="applet" code="MyApplet.class" width="400"
height="300">
</jsp:plugin>
```

Action `<jsp:param>`

Untuk applet kita dapat menetapkan satu atau lebih parameter. Misalnya, tag `<APPLET>` berikut ini memiliki beberapa parameter :

```
<APPLET CODE="MyApplet.class" WIDTH=400 HEIGHT=300>
  <PARAM NAME="param1" VALUE="value1">
  <PARAM NAME="param2" VALUE="value2">
</APPLET>
```

Atau dalam bentuk lain:

```
<APPLET CODE="MyApplet.class" WIDTH=400 HEIGHT=300>
  <jsp:params>
    <jsp:param NAME="param1" VALUE="value1" />
    <jsp:param NAME="param2" VALUE="value2" />
  </jsp:params>
</APPLET>
```

```
Action <jsp:fallback>
```

Dalam definisi applet kita dapat menetapkan teks alternatif jika browser gagal menjalankan applet.

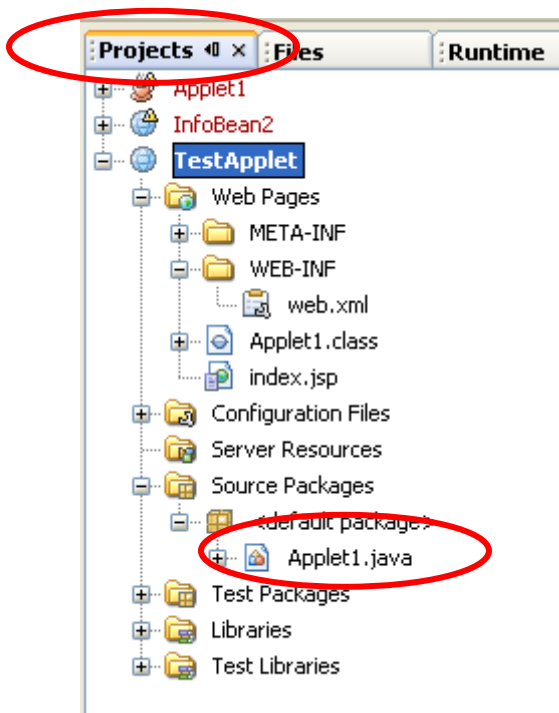
```
<APPLET CODE="MyApplet.class" WIDTH=400 HEIGHT=300>  
  Error:Applet ini memerlukan Java supaya dapat dijalankan  
</APPLET>
```

Dalam file JSP, teks alternatif dapat dituliskan di dalam action <jsp:plugin> dengan memakai action <jsp:fallback> sehingga contoh diatas dapat dituliskan sebagai berikut dalam format XML:

```
<jsp:plugin type="applet" code="MyApplet.class" width="400" height="300">  
  <jsp:fallback>  
    Error:Applet ini memerlukan Java supaya dapat di jalankan  
  </jsp:fallback>  
</jsp:plugin>
```

Contoh 1 : Project TestApplet

Buatlah project TestApplet, buatlah file applet sederhana masukkan dalam folder source packages berilah nama dengan Applet1.java (dengan default package). Struktur file pada project TestApplet adalah sebagai berikut:



Isi file Applet1.java

```
import java.applet.Applet ;  
import java.awt.*;
```

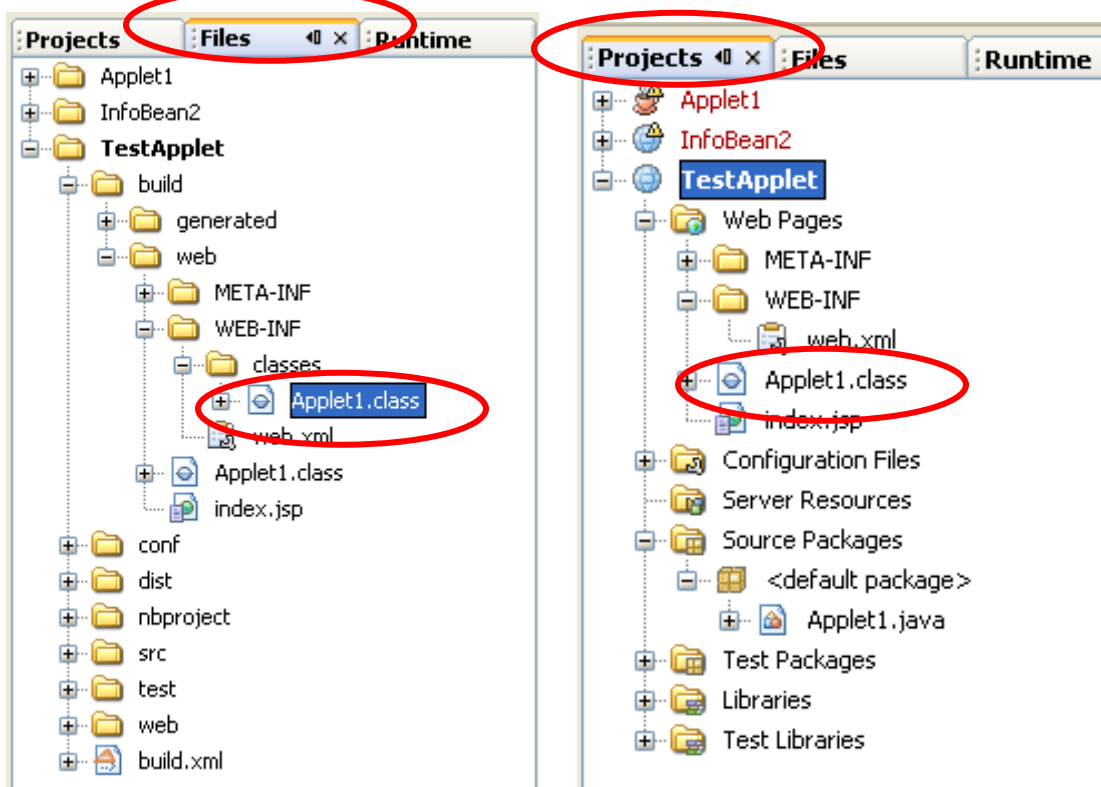
```

public class Applet1 extends Applet {
    /** Creates a new instance of Applet1 */
    public Applet1() {
        this.setBackground(Color.BLACK) ;
    }

    public void paint(Graphics g){
        g.setColor(Color.RED) ;
        g.fillOval(50,50,50,50);
    }
}

```

Selanjutnya lakukan compile pada Applet1.java, selanjutnya klik Files, lihat struktur file di bawah ini. Selanjutnya lakukan copy file Applet1.class masukkan pada Web Pages pada Project TestApplet.



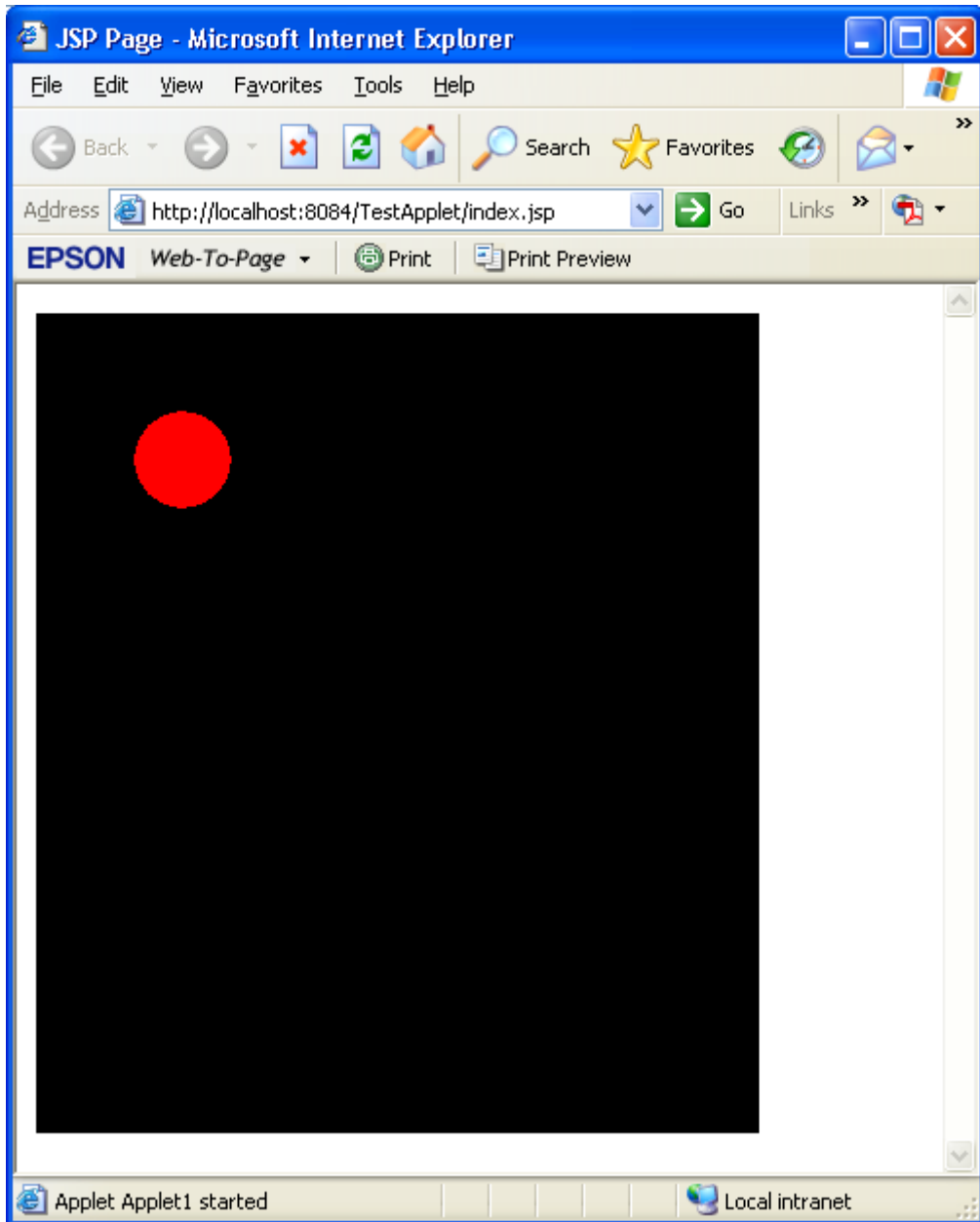
Isi dari file index.jsp

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <jsp:plugin type="applet"
      code="Applet1.class"
      width="370" height="420">
    </jsp:plugin>
  </body>
</html>

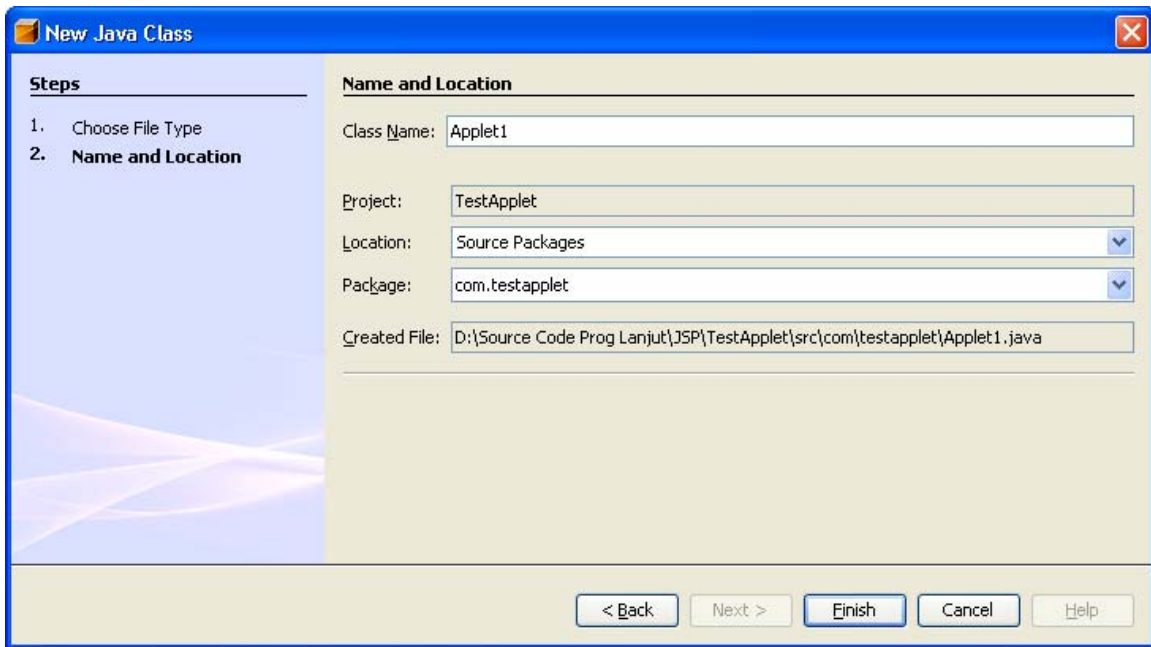
```

Selanjutnya lakukan compile dan build pada index.jsp. Hasilnya akan tampak sebagai berikut:



Bagaimana jika file applet berada pada package tertentu ?

Buatlah file Applet1.java dalam package com.testapplet. Cara membuatnya, pada source packages klik kanan, pilih java class. Selanjutnya akan muncul form seperti berikut:



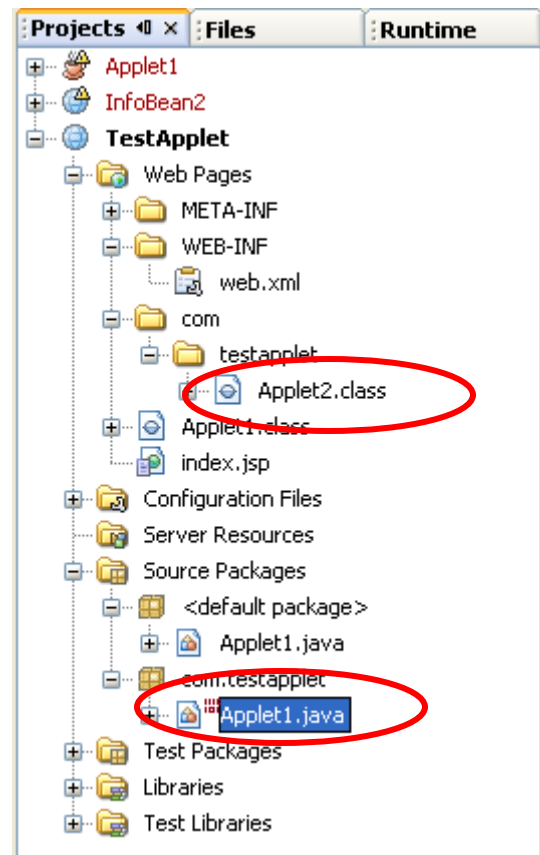
Applet1.java

```
package com.testapplet;
import java.applet.Applet ;
import java.awt.*;

public class Applet1 extends Applet {

    public Applet1() {
        this.setBackground(Color.BLACK);
    }

    public void paint(Graphics g){
        g.setColor(Color.RED) ;
        g.fillOval(50,50,50,50);
    }
}
```



Lakukan compile pada file Applet1. java pada package com.testapplet, selanjutnya copikan Applet1.classnya pada WebPages. Ingat yang dikopikan sekaligus dengan folder com. Ubah isi dari file index.jsp :

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <jsp:plugin type="applet"
      code="com.testapplet.Applet1.class"
      width="370" height="420">
    </jsp:plugin>
  </body>
</html>
```

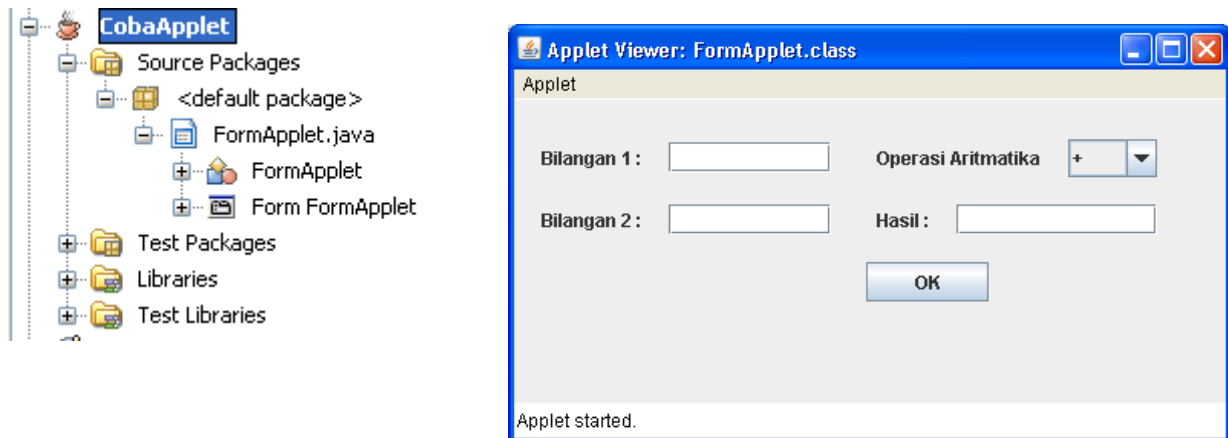
Contoh 2:

Bagaimana jika kita ingin menampilkan JApplet Form?
Buatlah dua project yaitu :

- Project untuk membuat form JApplet (CobaApplet)
- Project untuk Web Application (TestApplet).

Bagian 1:

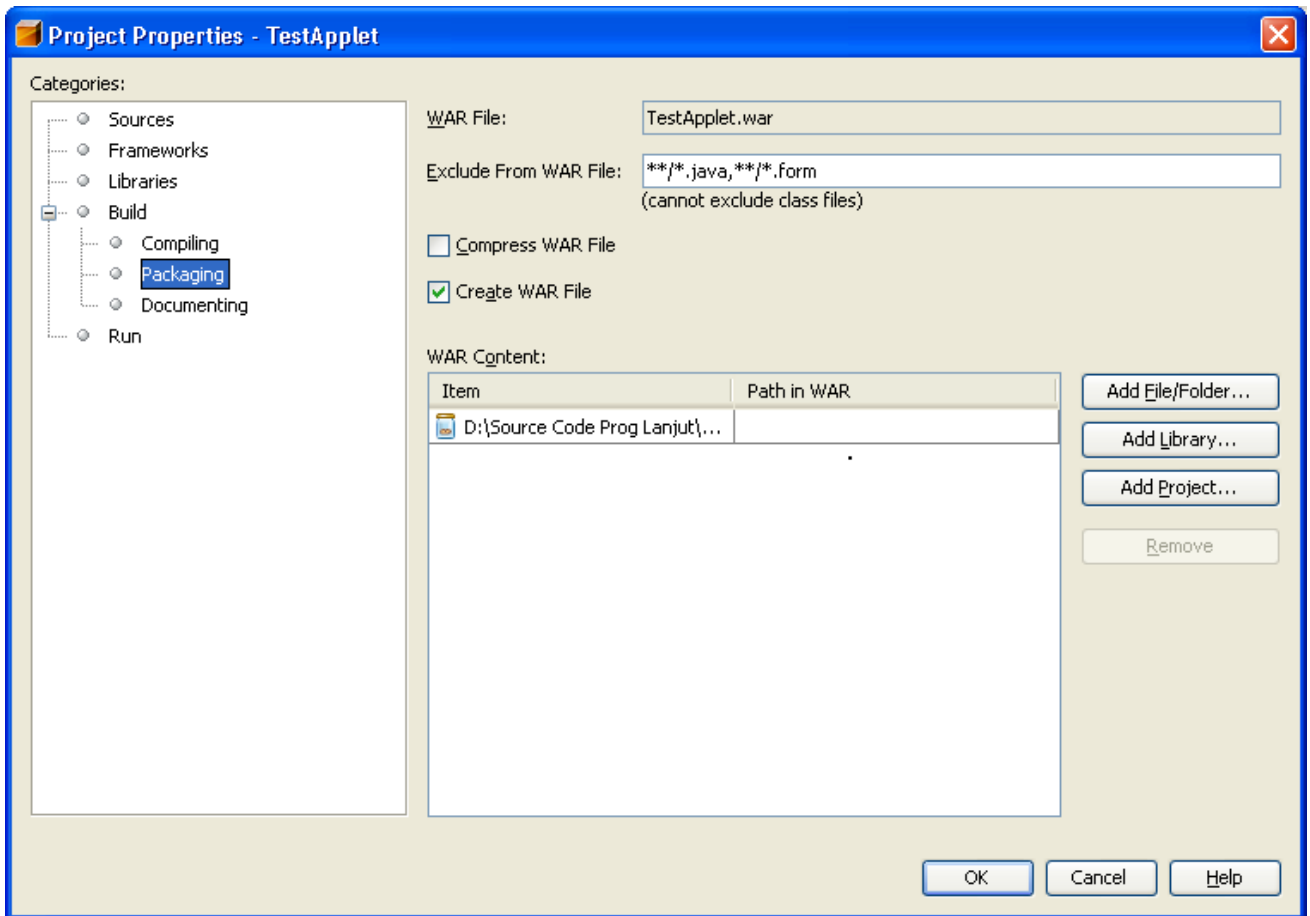
Buatlah form JApplet (beri nama dengan FormApplet.java) pada project CobaApplet, lakukan klik kanan pada project CobaApplet, jalankan Run Project maka hasil akan tampak sebagai berikut:



Selanjutnya lakukan build Project untuk file CobaApplet.jar yang berada pada folder dist.

Bagian 2 :

Klik kanan project TestApplet, pilih categories : Build → Packaging masukkan file jar (tekan button add file/folder), selanjutnya klik ok



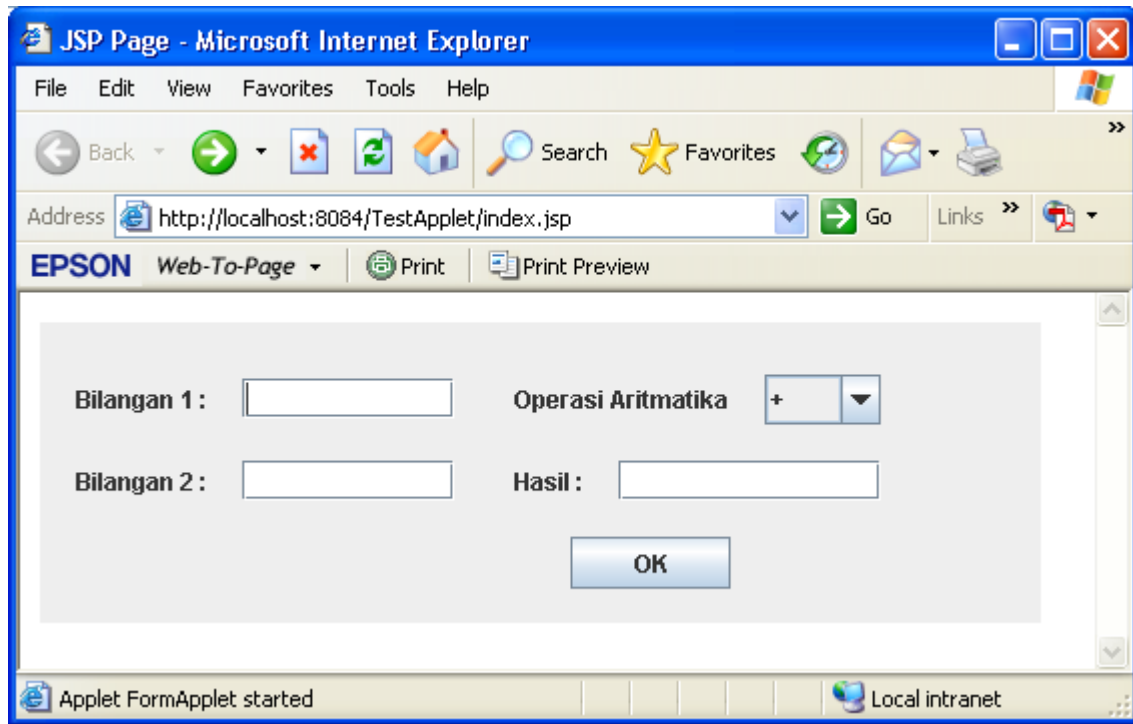
Ubah index.jsp

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <jsp:plugin type="applet"
      code="FormApplet.class" archive="CobaApplet.jar"
      width="500" height="150">
    </jsp:plugin>
  </body>
</html>

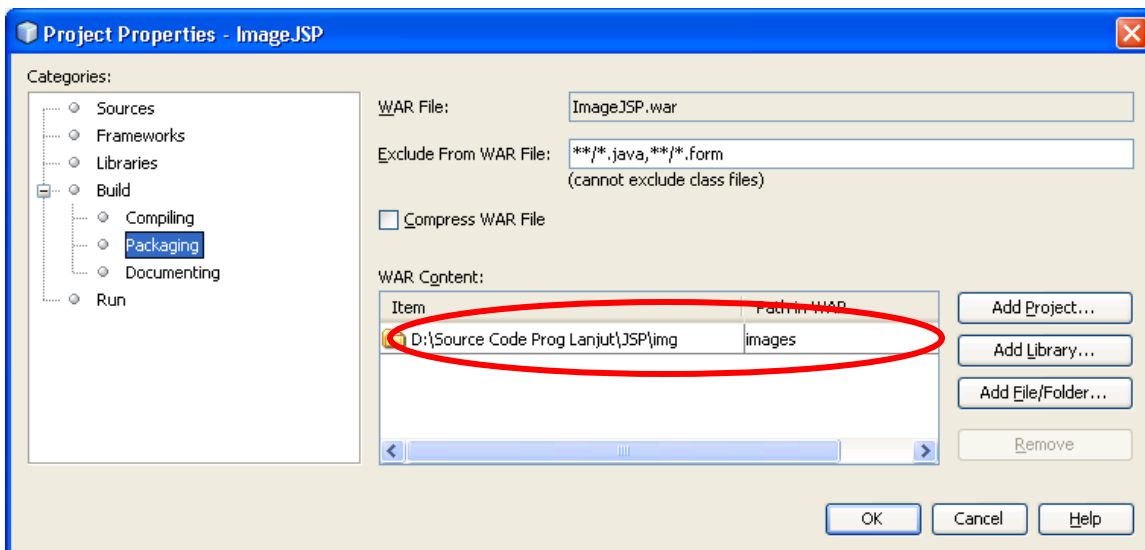
```

Hasil sebagai berikut:



JSP dan Image.

Untuk menampilkan gambar dalam halaman JSP, kita dapat memakai tag HTML ``. File-file gambar tambahkan pada Project Properties. Klik Add File/Folder masukkan folder img, beri nama path in WAR dengan "images".



Isilah file index.jsp dengan program berikut:

```
<html>
  <head>
```

```

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
    <table border="1">
    <tr> <td>
    <form method="GET" action="index.jsp">
    <select name="image" size="5">
        <option value="putih.jpeg" selected>Bunga Putih</option>
        <option value="red.jpeg">Bunga Merah</option>
        <option value="pink.jpeg">Bunga Pink</option>
        <option value="biru.jpeg">Bunga Biru</option>
        <option value="ungu.jpeg">Bunga Ungu</option>
    </select>
    <br>
    <input type="submit" value="submit" />
    </form>
    </td>

    <%

    String imgref = "images/" ;
    String param = request.getParameter("image") ;
    if (param==null)
        imgref += "putih.jpeg";
    else
        imgref += param;

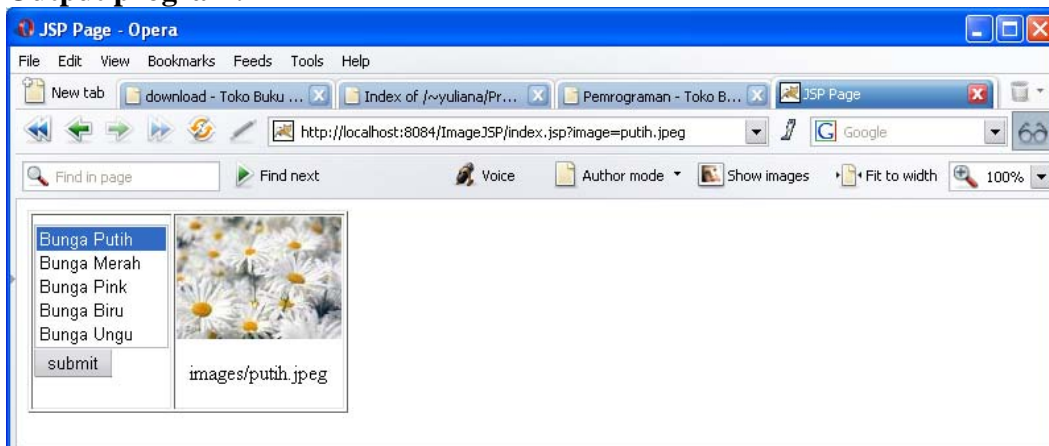
    %>

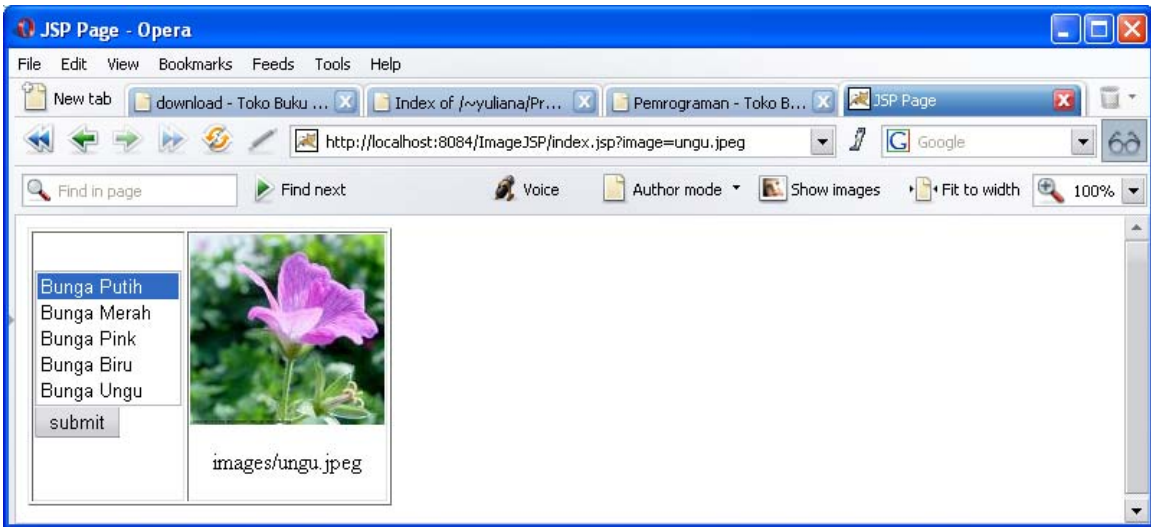
    <td>
        
        <p align="center"><%=imgref%></p>
    </td>
    </tr> </table>
    </body>
</html>

```

Selanjutnya lakukan build pada project ImageJSP, maka pada jendela Files akan ditampilkan file WAR berisi resource image yang berada pada folder dist\images(folder sesuai dengan nama path pada WAR).

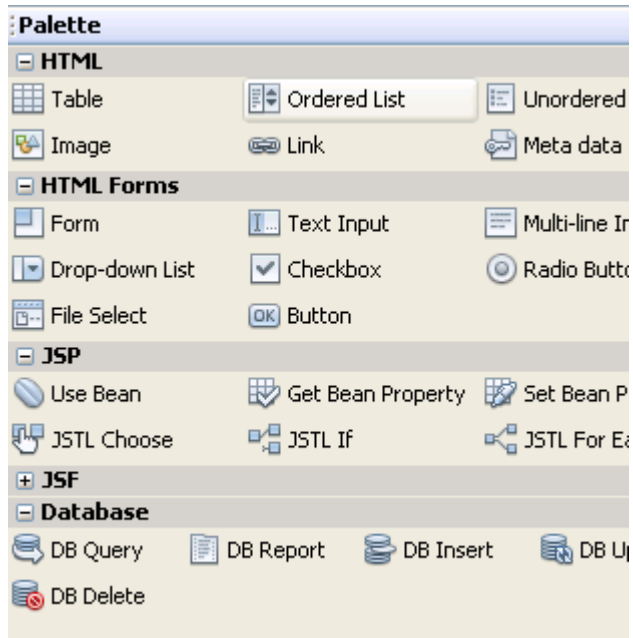
Output program:





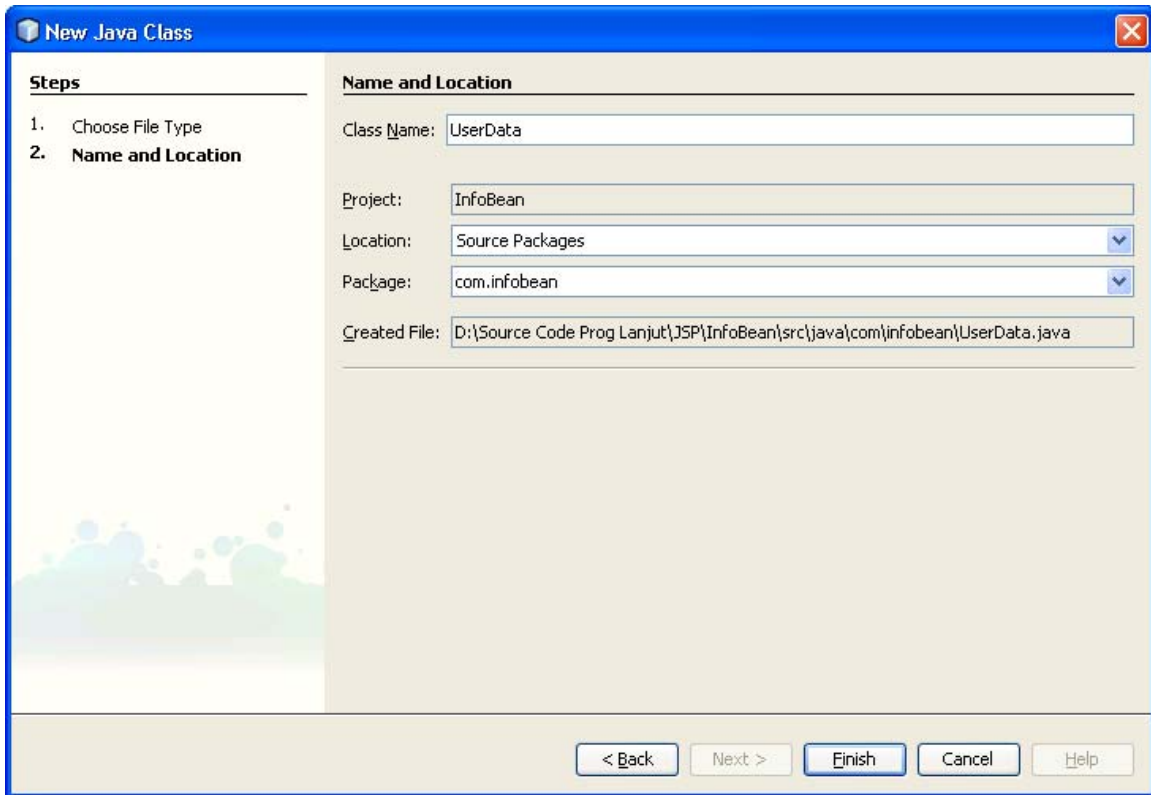
JAVA BEANS

JavaBeans adalah arsitektur standart untuk membuat komponen reusable dalam Java. Kita dapat membuat sebuah komponen (applet atau Java class) baik visual maupun non visual, mengubahnya dalam bentuk javabeans sehingga komponen tersebut dapat dipakai lagi kapanpun diinginkan. Jendela Pallette di IDE Netbeans sangat berguna untuk menambahkan elemen JSP, HTML, form atau database.



Contoh 1 :

- Buatlah project baru dengan nama InfoBean
- Dalam jendela project klik kanan Source Packages dan kemudian jalankan menu New File Classes. Isilah seperti gambar di bawah ini.



File UserData.java

```
package com.infobean;
public class UserData {
    // disebut sebagai properti
    // untuk masing2 properti disediakan method get sebagai getter dan
    // set sebagai setter.
    private String namaUser;
    private String idUser;
    private int umur ;
    public UserData(){
    }
    public String getNamaUser(){
        return namaUser;
    }
    public void setNamaUser(String namaUser){
        this.namaUser = namaUser;
    }
    public String getIdUser(){
        return idUser;
    }
    public void setIdUser(String idUser){
        this.idUser = idUser;
    }
    public int getUmur(){
        return umur;
    }
    public void setUmur(int umur){
        this.umur = umur;
    }
}
```



```
}  
}
```

Buatlah File Include/Segmen

Buatlah file include/segmen di folder WEB-INF/jspf dengan nama header.jspf

```
<%@ page import = "java.util.*, java.text.*" %>  
  
<%  
    DateFormat df = DateFormat.getDateInstance(DateFormat.FULL) ;  
    String sd = df.format(new Date()) ;  
%>  
  
<h4>Today: <%=sd%></h4><hr>
```

File index.jsp

```
<%@ include file="/WEB-INF/jspf/header.jspf" %>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>JSP Page</title>  
    </head>  
    <body>  
        <form name="login" action="sales.jsp" method="post">  
            <table width="200" border="1">  
                <tr>  
                    <td width="56">Nama </td>  
                    <td width="128"><input type="text" name="namaUser"></td>  
                </tr>  
                <tr>  
                    <td>ID</td>  
                    <td><input type="text" name="idUser"></td>  
                </tr>  
                <tr>  
                    <td>Umur</td>  
                    <td><input type="text" name="umur"></td>  
                </tr>  
                <tr>  
                    <td colspan="2"><input type="submit" name="Submit" value="Submit"></td>  
                </tr>  
            </table>  
  
        </form>  
    </body>  
</html>
```

File sales.jsp

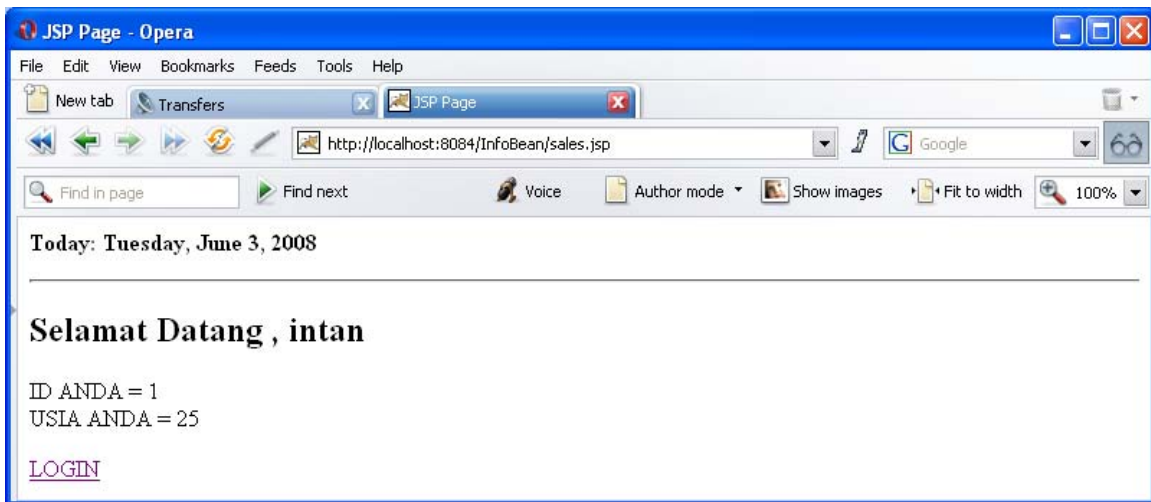
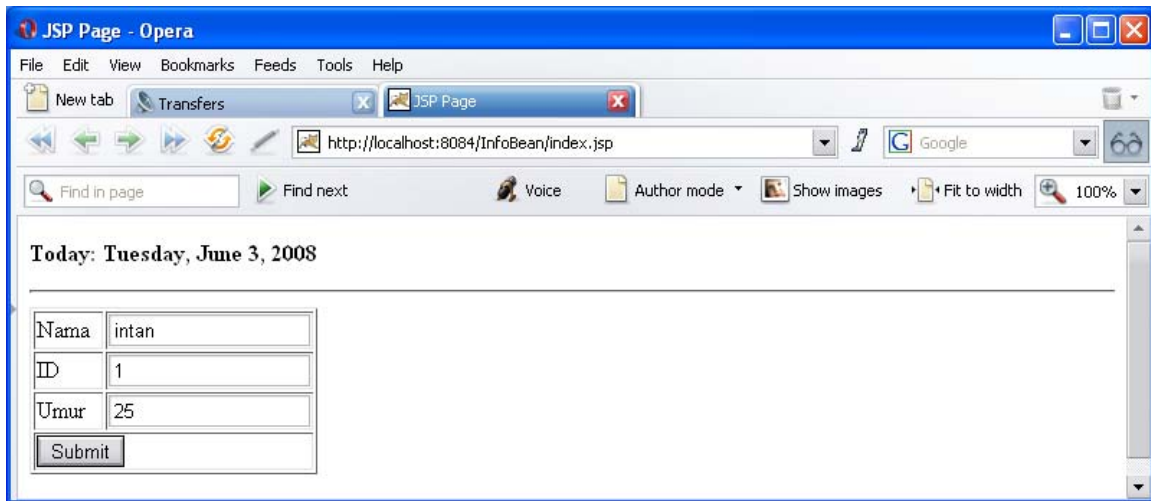
```
<%@ include file="/WEB-INF/jspf/header.jspf" %>  
  
<jsp:useBean id="userData" scope="session" class="com.infobean.UserData" />  
<jsp:setProperty name="userData" property="*" />
```

```

<html>
  <head>
    <title>JSP Page</title>
  </head>
  <body>
    <h2> Selamat Datang , <%=userData.getNamaUser()%> </h2>
    ID ANDA = <jsp:getProperty name="userData" property="idUser" /> <br>
    USIA ANDA = <jsp:getProperty name="userData" property="umur" /> <br>

    <p> </p>
    <a href="index.jsp" > LOGIN </a>
  </body>
</html>

```



Contoh 2 :

- Buatlah project baru dengan nama InfoBean2
- Dalam jendela project klik kanan Source Packages dan kemudian jalankan menu New File Classes. Buatlah class RandomNumber dalam package com.infobean.

RandomNumber.java

```
package com.infobean;
import java.util.Random;

public class RandomNumber {

    private int rndNumber2 ;
    public RandomNumber(){
        rndNumber2 = (int)(Math.random() * 100) ;
    }

    public int getRandomNumber(){
        return (int)(Math.random()*100) ;
    }

    public int getRndNumber2(){
        return rndNumber2 ;
    }

}
```

Buatlah file-file jsp dalam folder Web Pages. File header.jspf sama dengan contoh 1.

Random.jsp

```
<%@ include file="/WEB-INF/jspf/header.jspf" %>

<jsp:useBean id = "random" scope="request" class="com.infobean.RandomNumber" />

<html>
    <head>
        <title>JSP Page</title>
    </head>
    <body>
        Angka Acak ini menunjukkan tingkah laku Java Bean dengan scope =
request;
        <h3>
            <jsp:getProperty name="random" property="rndNumber2" />
        </h3>

        Angka Acak berikut ini menunjukkan method JavaBean tanpa memakai
property
        <h3>
            <%=random.getRandomNumber() %>
        </h3>

        Pilihan berikut ini akan membawa ke error page, jika Anda salah menebak.
        <form action="index.jsp" method="POST">
        <select name="language" size="4">
            <option value="java">JAVA</option>
            <option value="C" >C</option>
            <option value="Pascal" >PASCAL</option>
            <option value="Basic">BASIC</option>
        </select>
        <input type="submit" value="submit" />
        </form>
    </body>
</html>
```

index.jsp

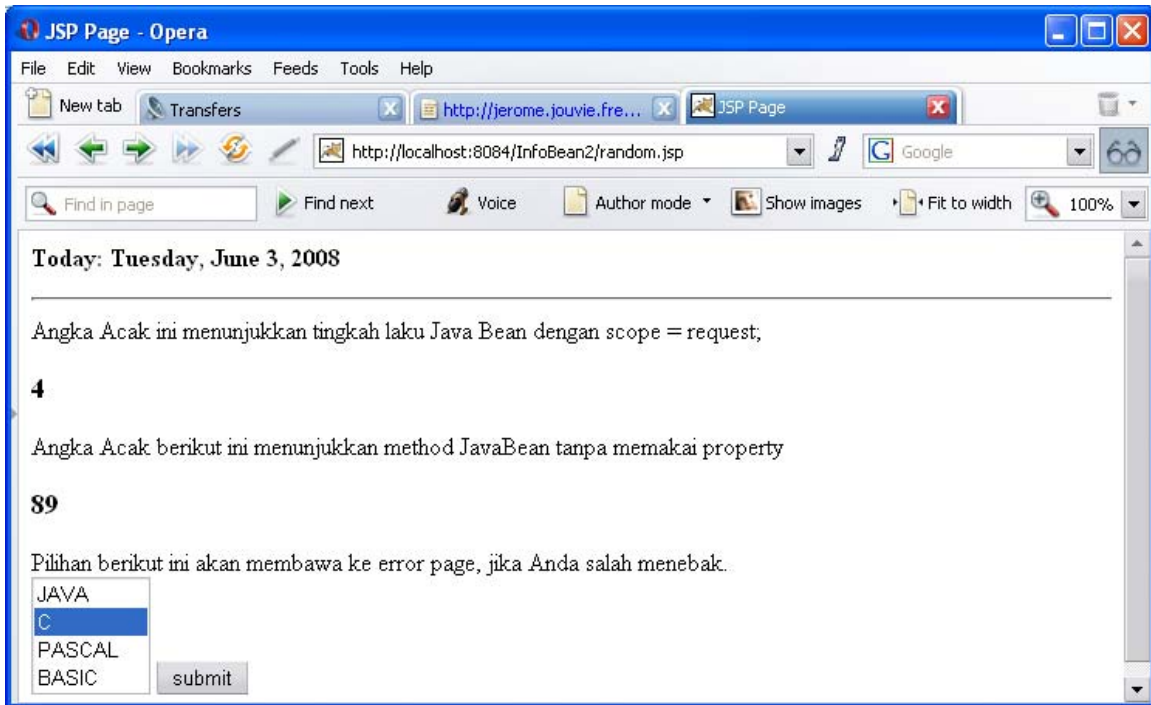
```
<%@ page errorPage="error.jsp" %>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%
      String lang = request.getParameter("language");
      if (lang.equalsIgnoreCase("java")){
    %>
    <h3> Pilihan Anda Benar ! (<%=lang%>) </h3>
    <% }
      else
      {
        //Pilihan salah, lemparkan exception
        throw new Exception("Pilihan Anda Salah !");
      }
    %>
  </body>
</html>
```

error.jsp

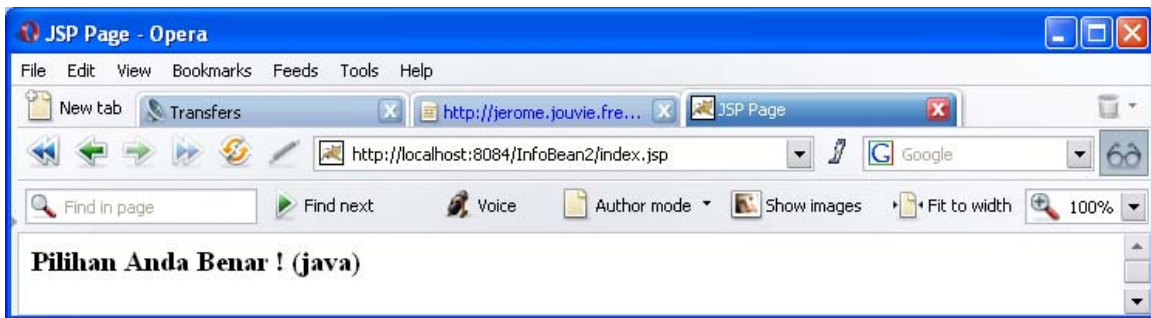
```
<%@ page isErrorPage="true" %>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1> Exception : </h1>
    <h3> <%=exception.getMessage() %>
  </body>
</html>
```

Output program:

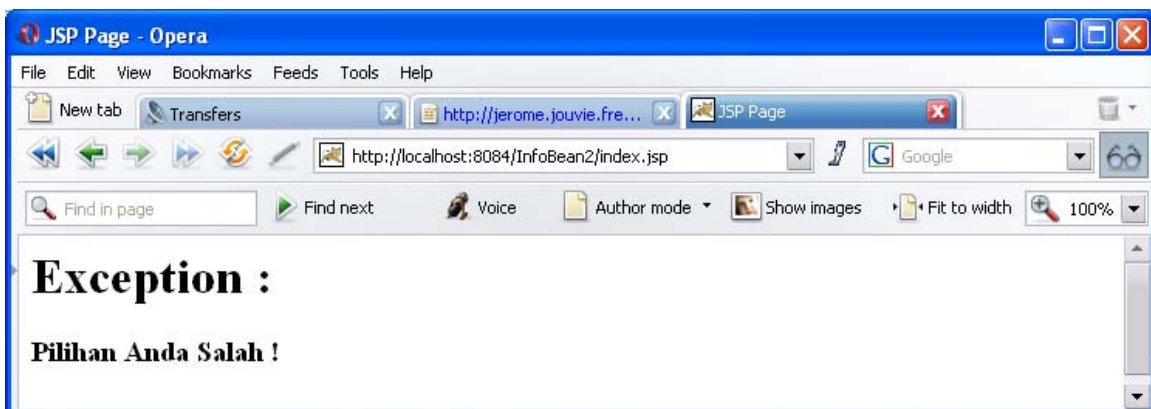
Pertama kali akan muncul gambar 1, jika kita pilih java maka akan muncul "Pilihan Anda Benar"(gambar 2) jika selain java akan melempar Exception (gambar 3).



Gambar 1



Gambar 2



Gambar 3

MODUL PRAKTIKUM:

Buatlah aplikasi web permainan tebak angka. Komponen Java Bean akan merahasiakan sebuah angka acak (1-100) dan user diminta untuk menebak angka tersebut. Jika tebakan lebih kecil, maka akan diberikan pesan agar user memasukkan angka yang lebih besar. Sebaliknya, jika tebakan lebih besar, user akan mendapatkan pesan agar memasukkan angka yang lebih kecil. Jumlah tebakan yang dilakukan user dihitung sampai mendapatkan tebakan yang benar.

Gunakan javabean dengan menggunakan scope session. Java Bean dengan jangkauan session memungkinkan aplikasi web untuk menyimpan data (atribut) yang dapat diakses antar halaman web.

Praktikum 1 :

Pertama kali aplikasi web seperti gambar 1. User diminta untuk memasukkan angka selanjutnya klik button tebak. Maka akan muncul peringatan lebih besar/ lebih kecil dari angka yang sebenarnya (gambar 2 dan gambar 3). Jika tebakan benar, maka hasil akhir ditunjukkan pada gambar 5.



Gambar 1



Gambar 2



Gambar 3



Gambar 4



Gambar 5

Komponen JavaBean

```
public class NumberGuessBean {
    private String guess ;
    private int numGuesses ;

    private String hint ;

    private boolean match ;

    private int answer ;

    public NumberGuessBean(){
        reset();
    }

    public void reset(){
        answer = (int) (Math.random() * 100) + 1;
    }
}
```



```

        match = false ;
        numGuesses = 0 ;
    }
    public String getGuess(){
        return this.guess ;
    }

    public void setGuess(String guess){
        if (guess !=null) numGuesses++;

        int gs ;
        try{
            gs = Integer.parseInt(guess) ;
        }
        catch(NumberFormatException e){
            gs=-1 ;
        }

        if (gs==-1)
            hint = "Error : Isikan hanya angka digit" ;
        else if (gs < answer)
            hint = "Lebih besar .... Tebakan terlalu kecil" ;
        else if (gs > answer)
            hint = "Lebih kecil .... Tebakan terlalu besar" ;
        else if (gs == answer)
        {
            hint = "Selamat ! Tebakan Anda Benar...." ;
            match = true;
        }
        this.guess = String.valueOf(gs) ;
    }

    public int getNumGuesses(){
        return this.numGuesses ;
    }

    public String getHint(){
        return this.hint ;
    }
    public boolean isMatch(){
        return this.match ;
    }
}

```

Terdapat 4 property

- **Guess:** untuk menyimpan angka tebakkan dari user. Tipe dipilih string, karena user dapat saja mengetikkan karakter non-digit.
- **Hint :** dipakai untuk menyimpan petunjuk yang akan ditampilkan ke user. Jika angka tebakkan dari user lebih kecil dari jawaban maka akan ditampilkan hint="Lebih besar...". Mode di set readonly sebab nilainya tidak boleh di set dari luar.
- **numGuess :** dipakai untuk menyimpan jumlah tebakkan yang telah dilakukan oleh user. Mode adalah readonly sebab jumlah tebakkan akan dihitung otomatis dan tidak boleh diubah dari luar.
- **Match :** dipakai untuk menunjukkan apakah user telah menebak dengan benar atau tidak. Mode readonly karena nilainya diset benar oleh internal bean hanya setelah user menebak dengan benar.

Dalam class JavaBean ditambahkan field "answer" untuk menyimpan angka acak yang akan ditebak kemudian ditambahkan method reset() untuk menentukan angka acak dan melakukan inisialisasi. Angka acak diset antara 1-100. Dibawah ini adalah kerangka dari file index.jsp. Selamat mencoba.

index.jsp

```
//lakukan import class NumberGuessBean yang terletak pada package
com.jsp

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>

    <!-- buatlah sebuah objek numGuess dengan scope session dengan
menggunakan javabean --%>

    <!-- lakukan pengesetan untuk semua property dengan melakukan
property="*" --%>

    <h1><u> Number Guess Game </u></h1>

    <h3>
      <!-- Jika mencoba pertama kali maka --%>

      Silakan tebak sebuah bilangan antara 1 s/d 100.
      <!-- Buatlah sebuah form dengan method="GET" dan
action="index.jsp". Beri nama textfield dengan guess dan button submit.
--%>

      <!-- Jika tidak (bukan pertama kali) maka lakukan --%>

      <!-- Jika tebakan sama dengan angka acak yang dibangkitkan
maka
          Tampilkan tebakan dari user
          Tampilkan petunjuk untuk user (method getHint())
          Tampilkan informasi user mencoba berapa kali.
          Lakukan method reset(untuk melakukan inisialisasi dan
melakukan pembangkitan acak lagi)
--%>
      <p> Silakan <a href="index.jsp"> COBA LAGI </a> </p>

      <!--Jika tebakan tidak sama maka
          Tampilkan tebakan dari user
          Tampilkan petunjuk untuk user (method getHint())
          Tampilkan informasi user mencoba berapa kali. --%>
      <br>
      Silakan tebak lagi bilangan 1 s/d 100
      <!-- Buatlah sebuah form dengan method="GET" dan
action="index.jsp". Beri nama textfield dengan guess. Dan buatlah
button submit--%>
```

```
</h3>
</body>
</html>
```

Praktikum 2: GuestBook

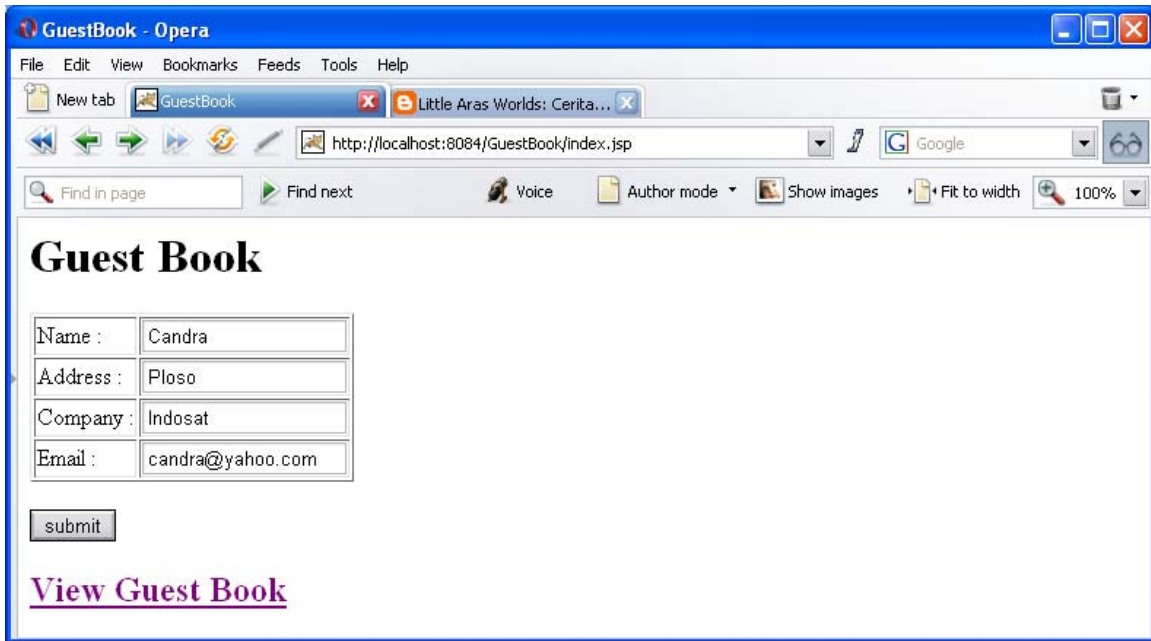
Buatlah aplikasi web GuestBook, dimana user memasukkan data nama, alamat, perusahaan dan email. Data ini selanjutnya akan tersimpan pada database access (GuestBook). Terdapat sebuah table GuestBook dengan field name, address, company dan email, semuanya bertipe String. Aplikasi ini menggunakan JavaBean untuk melakukan koneksi ke database, menyisipkan data dan untuk menampilkan data(GuestBookBean.java).

Terdapat 3 file yaitu index.jsp, GuestBook.jsp dan GuestBookView.jsp.

Nama File	Keterangan
Index.jsp	Halaman utama dari aplikasi. Pada halaman ini user dapat mengisi GuestBook dengan memasukkan nama, alamat, perusahaan bekerja dan email.
GuestBook.jsp	Halaman yang memproses data user untuk diinsertkan ke database.
GuestBookView.jsp	Halaman untuk menampilkan semua user yang tersimpan pada tabel GuestBook.

Aplikasi :

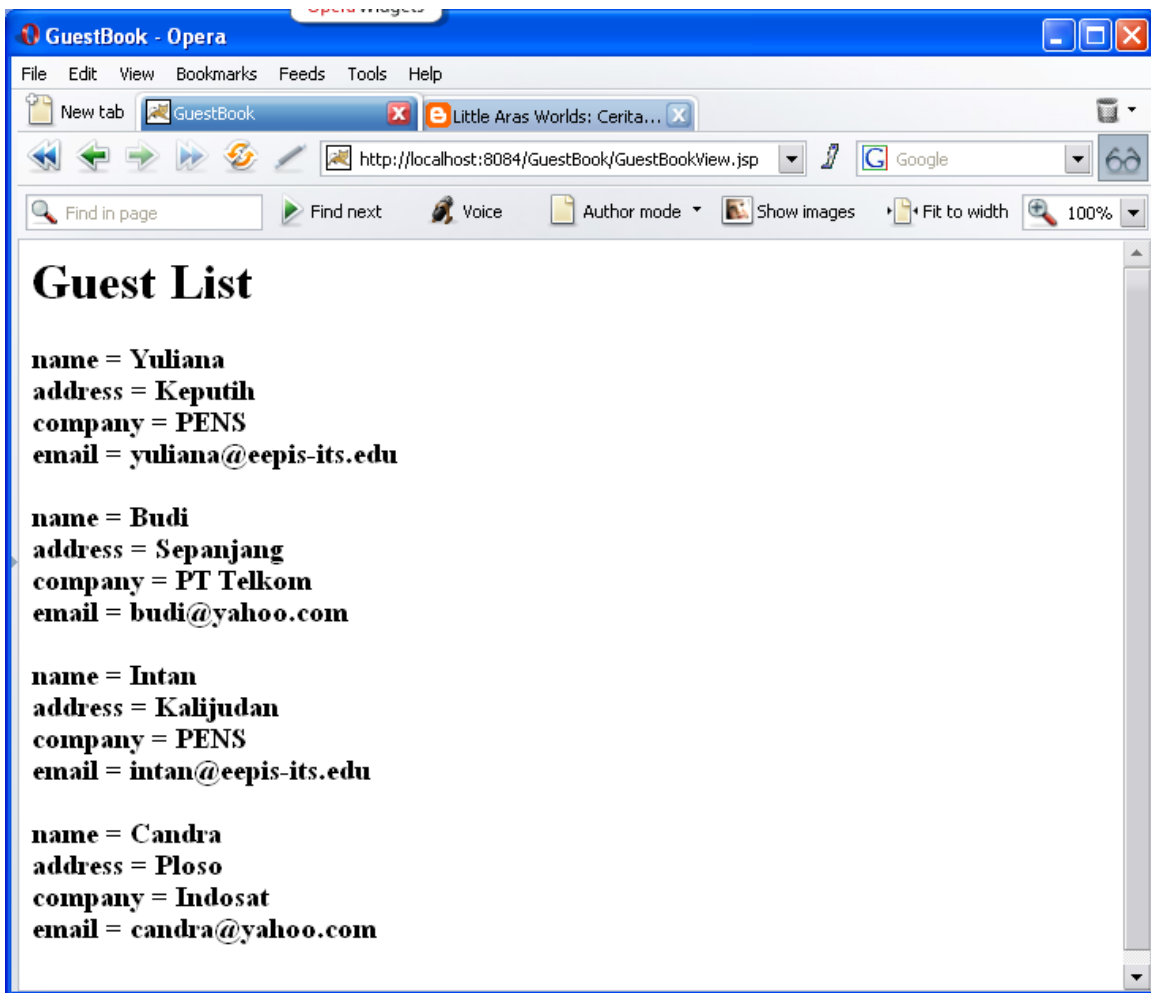
Pertama kali user memasukkan nama, alamat, perusahaan dan email (gambar1). Selanjutnya klik button submit, jika berhasil akan muncul pesan "Thank you Candra for registering"(gambar 2). Selanjutnya pilih View Guest Book, maka akan menampilkan data semua user (gambar 3).



Gambar 1



Gambar 2



Gambar 3

class GuestBookBean

```
package guestBook;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Vector;

public class GuestBookBean {
    private Statement stmt=null ;
    private Connection conn = null ;
    private String sURL = "jdbc:odbc:GuestBook" ;
    private ResultSet rs=null ;

    public boolean insertIntoDB (String name, String address, String company,
String email){
        String sql = "INSERT INTO GuestBook
VALUES('"+name+"','"+address+"','"+company+"','"+email+"') " ;
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
            conn = DriverManager.getConnection(sURL,"","") ;
            stmt=conn.createStatement() ;
            stmt.execute(sql);
            stmt.close();
            return true;
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }
    }

    public Vector viewGuestBook(){
        Vector vGuest = new Vector();
        String sData;
        String sql = "SELECT * FROM GuestBook" ;
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
            conn = DriverManager.getConnection(sURL,"","") ;
            stmt=conn.createStatement() ;
            rs = stmt.executeQuery(sql) ;

            while(rs.next()){
                sData = "name = " + rs.getString(1) + "<br>address = " +
rs.getString(2) + "<br>company = " + rs.getString(3) + "<br>email = " +
rs.getString(4);
                vGuest.add(sData);
                sData="";
            }
            rs.close();
            stmt.close();
        }catch(Exception e){
            e.printStackTrace();
        }
        return vGuest;
    }
}
```

index.jsp

```
<html>
  <head>
    <meta      http-equiv="Content-Type"      content="text/html;
charset=UTF-8">
    <title>GuestBook</title>
  </head>
  <body>
    <h1>Guest Book</h1>
    <!-- Buatlah sebuah form yang menerima masukkan user nama,
alamat, perusahaan dan email. Beri nama textfield dengan nama name,
address, company dan email. Isi form ini dengan action="GuestBook.jsp"
method="post". -->

    <h2><a href="GuestBookView.jsp" > View Guest Book </a></h2>
  </body>
</html>
```

GuestBook.jsp

```
<!-- definisikan sebuah objek JavaBean dengan name = guestbook dari
class guestBook.GuestBookBean dengan scope=page -->

<%
String message = "";
String name = request.getParameter("name") ;
String address = request.getParameter("address") ;
String company = request.getParameter("company") ;
String email = request.getParameter("email") ;

//jika name, address, company dan email tidak ("") maka
//lakukan insert data ke database
//jika berhasil message = "Thank you " + name + " for Registering "
;
//jika tidak set message = "Error" ;
%>

<html>
  <head>
    <meta      http-equiv="Content-Type"      content="text/html;
charset=UTF-8">
    <title>GuestBook</title>
  </head>
  <body>
    <h2><%=message%>
    <br>
    <a href="index.jsp"> GUEST BOOK </a>
    <br>
    <a href="GuestBookView.jsp"> VIEW GUEST BOOK </a>
  </h2>
  </body>
</html>
```

GuestBookView.jsp

```
<!-- lakukan import class Vector.
definisikan sebuah objek JavaBean dengan name = guestbook dari class
guestBook.GuestBookBean dengan scope=page --%>

<%Dapatkan data semua user (simpan pada Vector vGuest) dengan memanggil
fungsi viewGuestBook(); %>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>GuestBook</title>
  </head>
  <body>
    <h1>Guest List </h1>
    <h3>
      <%
        //lakukan iterasi pada vGuest dan tampilkan hasilnya
      %>
    </h3>
  </body>
</html>
```

Praktikum 3: Pengembangan dari GuestBook

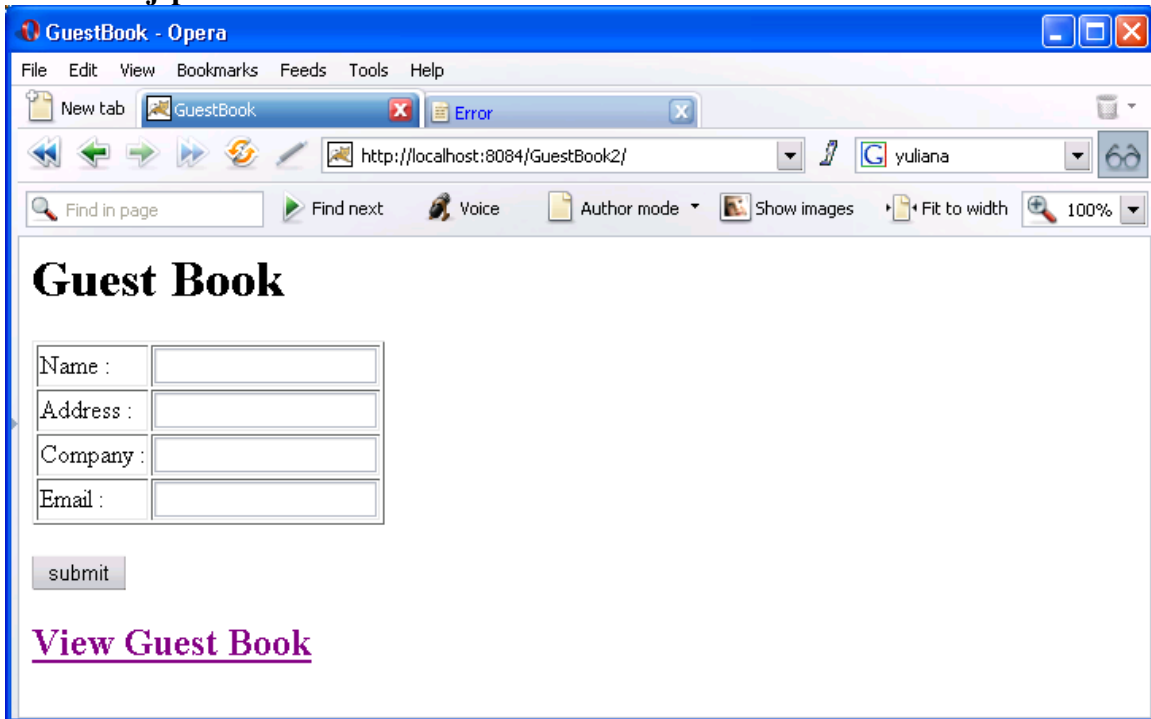
Kembangkan dari praktikum 2, tambahkan proses edit dan delete. Aplikasi utama ditunjukkan pada gambar 1. Klik View Guest Book, hasil seperti gambar 2. Terdapat 3 user yang telah terdaftar yaitu Yuliana, Budi dan Intan. Masing-masing user terdapat fasilitas untuk mengedit dan menghapus. Klik edit pada user Intan dengan alamat Kalijudan, lakukan pengeditan data pada alamat ubah menjadi Kalijudan 57, selanjutnya lakukan submit (gambar 4). Pada gambar 4 klik View Guest Book maka hasil seperti gambar 5.

Pada user Intan lakukan delete, maka hasil tampak seperti gambar 6, selanjutnya klik View Guest Book maka hasil seperti gambar 7.

Terdapat file-file pendukung yaitu:

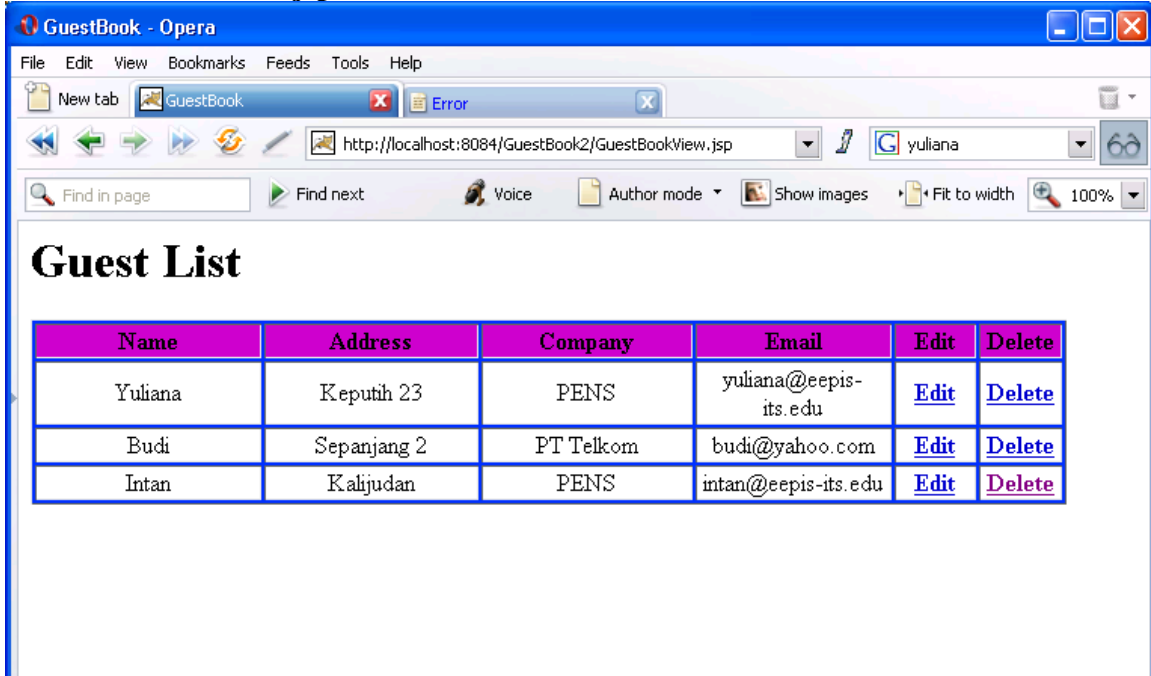
Nama File	Keterangan
Index.jsp	Halaman utama dari aplikasi. Pada halaman ini user dapat mengisi GuestBook dengan memasukkan nama, alamat, perusahaan bekerja dan email.
GuestBook.jsp	Halaman yang memproses data user untuk diinsertkan ke database.
GuestBookView.jsp	Halaman untuk menampilkan semua user yang tersimpan pada tabel GuestBook.
GuestBookEdit.jsp	Halaman ini menampilkan data user yang akan diedit.
Prosesedit.jsp	Halaman yang memproses data user untuk diupdate di database
Delete.jsp	Halaman yang memproses data user untuk dihapus di database

File index.jsp



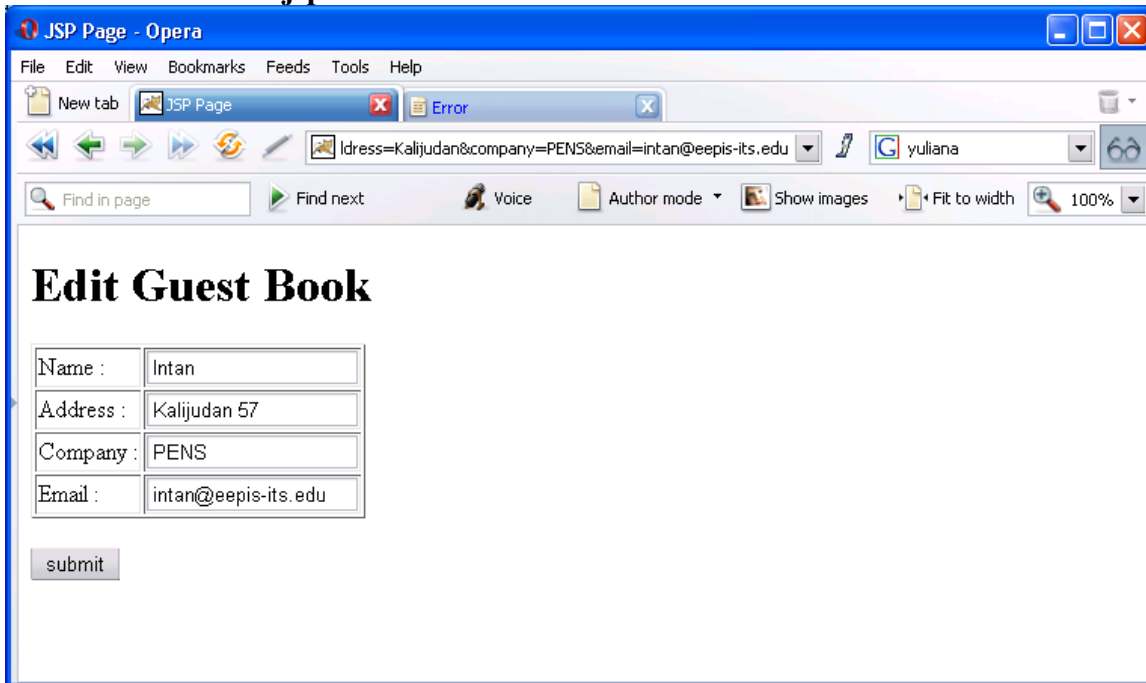
Gambar 1

File GuestBookView.jsp



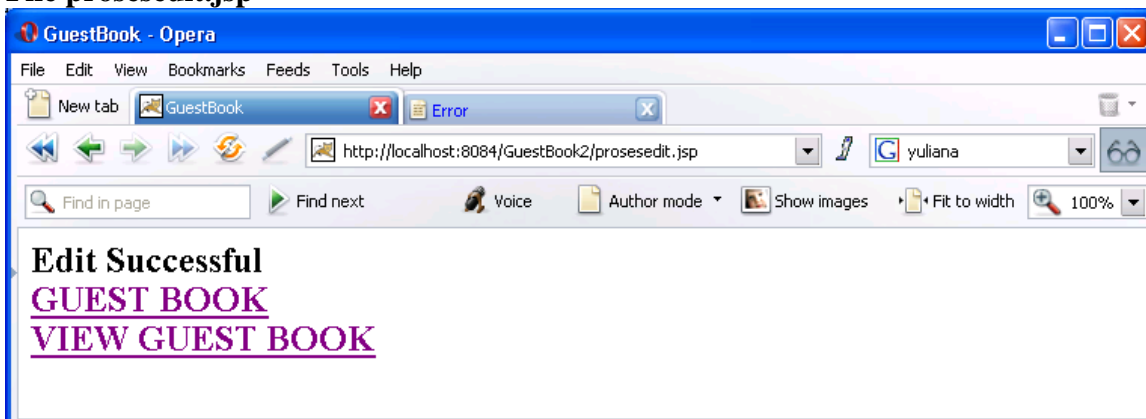
Gambar 2

File GuestBookEdit.jsp



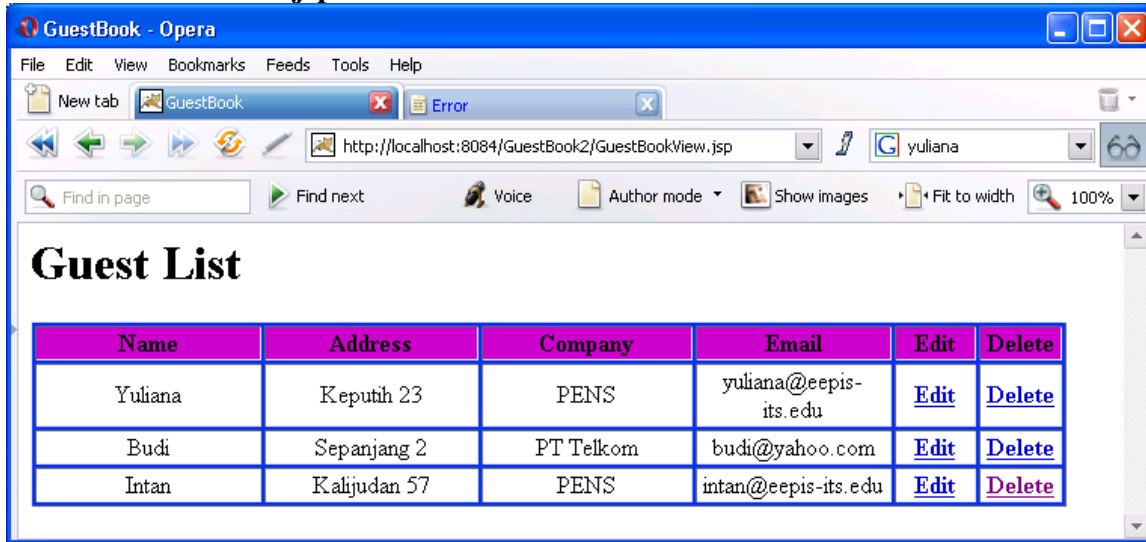
Gambar 3

File prosesedit.jsp



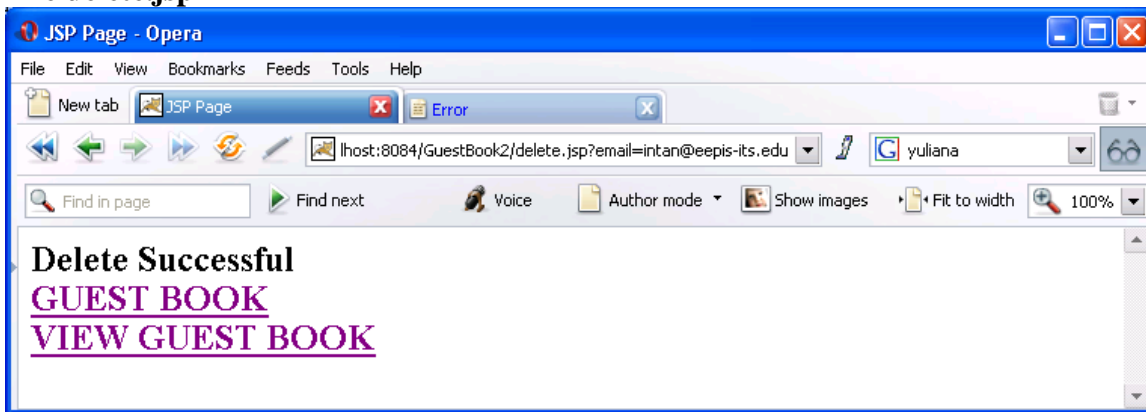
Gambar 4

File GuestBookView.jsp



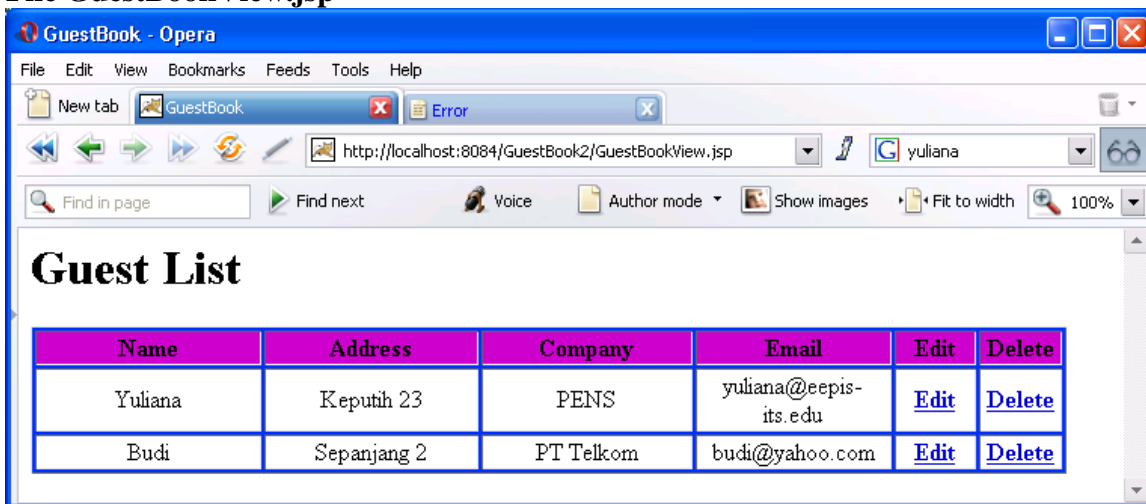
Gambar 5

File delete.jsp



Gambar 6

File GuestBookView.jsp



Gambar 7

GuestBookView.jsp

```
<%----%>

<%-- import class Vector --%>
<%-- buatlah sebuah object javabeen dari class GuestBookBean pada package
guestBook dengan scope page --%>

<%-- panggil method viewGuestBook simpan hasilnya dalam Vector dengan nama
vGuest --%>

<html>
  <body>
    <h1>Guest List </h1>
    <h3>
      <%-- Memasukkan user tamu dalam tabel--%>

      <table>
        <%-- Baris header table yaitu name, address, company, email, edit dan
delete --%>
        <tr> <td> Name</td>
          <td> Address</td>
          <td> Company</td>
          <td> Email</td>
          <td> Edit</td>
          <td> Delete</td>
        </tr>
        <%
          for(int i=0;i<vGuest.size();i++){
            /*ambil data pada Vector pada indek ke-1(data berupa vector (nama
temp)).
            Ambil nama user (simpan dalam variabel nm dengan tipe String),
address (simpan dalam variabel adr dengan tipe String), company( simpan dalam
variabel comp dengan tipe String) dan email(simpan dalam variabel email dengan
tipe String). */

            %>
            <tr> <td> <%=nm%></td>
              <td> <%=adr%></td>
              <td><%=comp%></td>
              <td> <%=email%></td>
              <td><a
href="GuestBookEdit.jsp?name=<%=nm%>&address=<%=adr%>&company=<%=comp%>&email=<
%=email%>">Edit</a></td>
              <td> <a href="delete.jsp?email=<%=email%>">Delete</a> </td>
            </tr>

            <%
              }
            %>
          </table>
          </h3>
        </body>
      </html>
```

Membuat Toko Buku Sederhana (Bagian dari E-Commerce)

Pada toko buku ini, pelanggan bisa melihat-lihat buku yang ada pada katalog, kemudian melakukan pencarian buku berdasarkan keyword yang dapat dimasukkan melalui search box yang ada pada setiap halaman katalog. Pada halaman depan pemilik toko buku bisa menampilkan buku-buku yang ingin dipromosikan secara khusus.

Setelah menemukan buku yang tepat, pelanggan dapat memasukkan buku yang diinginkan ke keranjang belanjanya (shopping cart). Pelanggan bisa kembali melihat-lihat katalog dan menambahkan buku lain ke keranjang belanjanya. Pelanggan juga dapat melakukan update terhadap jumlah buku yang ingin dibeli dan bisa membatalkan pembelian dengan cara mengeluarkan buku yang ada dari keranjang belanjanya.

Database

Terdapat 3 tabel yang digunakan :

Nama Tabel	Keterangan
Books	Tabel berisi data-data buku yang ada
Categories	Tabel untuk menyimpan tipe kategori dari buku-buku yang ada
Promotion	Tabel yang menyimpan data buku yang ingin ditampilkan pada halaman depan/dipromosikan.

Penjelasan masing-masing tabel.

Tabel Books

Nama field	Tipe data
product_id	text
title	text
author	text
description	text
price	int
category_id	text
image	text

Tabel Categories

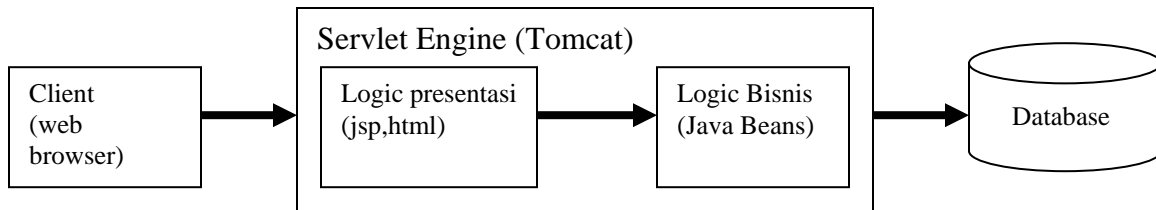
Nama field	Tipe data
category_id	text
category_name	text
category_description	text

Tabel Promotion

Nama field	Tipe data
promotion_id	text
product_id	text

Arsitektur Toko Buku Sederhana

Bagian ini menjelaskan tentang arsitektur sistem yang akan digunakan.



Suatu web-store biasanya terdiri atas beberapa tier atau lebih sering disebut dengan n-tier application. Suatu aplikasi n-tier biasanya terdiri atas beberapa layer/lapisan. Layer pertama berisi logika presentasi yaitu suatu layer yang berisi logika untuk menampilkan data yang diinginkan oleh client. Layer kedua berisi logika bisnis yang melakukan proses penerapan dari aturan-aturan bisnis yang ada dan layer ketiga berupa database.

Pada suatu aplikasi web yang membutuhkan skalabilitas yang lebih tinggi, maka logika bisnis yang ada dapat dipisahkan lagi pada tier lain dan logika bisnis dibuat sebagai komponen Enterprise Java Bean.

Logika presentasi.

Pemisahan antara logika presentasi dan logika bisnis akan dilakukan dengan memisahkan program Java, yang bisa ditempatkan pada halaman JSP ke dalam suatu class Java yang akan dipergunakan oleh JSP sebagai bean. Pada logika presentasi ini, program java yang ada pada halaman-halaman JSP secara khusus akan berfungsi untuk menampilkan data sesuai request dari client.

Nama File	Keterangan
header.html	File html statis untuk menampilkan header yang berisi logo. File ini akan diikutkan pada seluruh halaman JSP yang lain.
index2.jsp	Halaman utama yang berfungsi untuk menampilkan halaman yang berisi buku-buku yang dipromosikan.
index3.jsp	Halaman yang menampilkan kategori buku
catalog.jsp	Halaman yang berfungsi untuk menampilkan katalog yang berisi buku-buku tertentu sesuai keyword yang dimasukkan.
shopcart.jsp	Halaman JSP yang menampilkan shopping cart dari seorang pelanggan. Pada halaman ini pelanggan dapat menghapus pesanan yang tidak diinginkan.

Logika Bisnis

Logika bisnis ini terdiri atas beberapa class java yang akan digunakan oleh halaman JSP dengan action tag `<jsp:useBean>` untuk menampilkan data. Class java yang ada ini mempunyai fungsi khusus yang berkaitan dengan logika bisnis dimana mereka berada.

Nama File	Keterangan
CMbBooks.java	Suatu class java yang merepresentasikan objek buku
CatalogBean.java	Suatu class java yang berisi method-method yang berkaitan dengan katalog. Misal menampilkan buku berdasarkan

	keyword yang diinginkan
shoppingCartBean.java	Suatu class java yang berisi method-method yang berkaitan dengan shopping cart. Misal menambahkan barang ke dalam shopping cart, mengubah jumlah barang dan menghapus barang dari shopping cart.

PEMBAHASAN PROGRAM

CmbBooks.java

```
package com.jsp;
```

```
public class CmbBooks {
    private String product_id ;
    private String title ;
    private String author;
    private String description ;
    private int price ;
    private String image ;

    public void setProduct_id(String product_id){
        this.product_id = product_id ;
    }

    public String getProduct_id(){
        return this.product_id ;
    }

    public void setTitle(String title){
        this.title = title ;
    }

    public String getTitle(){
        return this.title ;
    }

    public void setAuthor(String author){
        this.author = author ;
    }

    public String getAuthor(){
        return this.author ;
    }

    public void setDescription(String description){
        this.description = description ;
    }

    public String getDescription(){
        return this.description ;
    }
}
```

```

    public void setPrice(int price){
        this.price = price ;
    }

    public int getPrice(){
        return this.price ;
    }

    public void setImage(String image){
        this.image = image ;
    }

    public String getImage(){
        return this.image ;
    }

    public String toString(){
        return product_id+ " " + title;
    }
}

```

Penjelasan :

Objek yang dihasilkan dari class CmbBooks merupakan pemetaan dari suatu buku yang ada pada satu baris di tabel Books pada database. Objek dari class CmbBooks ini akan digunakan oleh objek-objek lain yang melakukan pemrosesan terhadap objek buku. Misalnya pada objek catalog, objek catalog ini akan menginstance sejumlah objek books sesuai dengan kategori tertentu dan meletakkan objek-objek ini pada vector untuk ditampilkan pada halaman JSP>

Katalog

Komponen terpenting berikutnya bagi suatu aplikasi web-store adalah katalog, dimana dengan menggunakan katalog pelanggan bisa melakukan pencarian terhadap produk yang ingin dibeli, serta melihat-lihat keterangan mengenai product yang ada. Untuk membuat suatu katalog bagi web-store yang bersifat dinamis, suatu aplikasi harus mengakses database dan menampilkan data-data produk yang ada ke halaman katalog. Pada aplikasi ini, seluruh fungsi yang berkaitan dengan katalog didefinisikan pada file CatalogBean.java

CatalogBean.java

```

package com.jsp;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Vector;

public class CatalogBean {
    private Statement stmt=null ;
    private Connection conn = null ;
    private String sURL = "jdbc:odbc:BookStore" ;
    private ResultSet rs=null ;
}

```



```

public boolean connect(){
    try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;
        conn = DriverManager.getConnection(sURL,"","") ;
        return true ;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

public boolean disconnect(){
    try{
        stmt.close();
        rs.close();
        return true ;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

//untuk mengambil data kategori buku (digunakan)
public Vector getAllCatalog(){
    Vector vCategory = new Vector();
    Vector temp ;
    String sql ;

    try{
        sql="SELECT * FROM Categories";

        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        while (rs.next()) {
            temp = new Vector();
            temp.add(rs.getString(1));
            temp.add(rs.getString(2));
            vCategory.add(temp);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    finally{
        disconnect();
    }
    return vCategory ;
}

//mengambil judul katalog berdasarkan categoryID
public String getCategoryName(String category_ID){
    String sql ;
    String sCategoryName="" ;
    try{
        sql="SELECT          category_name          FROM          Categories          WHERE
category_id='"+category_ID+"' " ;

        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        if (rs.next()) {

```

```

        sCategoryName = rs.getString("category_name") ;
    }
    catch(Exception e){
        e.printStackTrace();
    }
    finally{
        disconnect();
    }
    return sCategoryName ;
}

```

//mengambil data buku dari database berdasarkan kategori

```

public Vector getBooksCatalog(String category_ID){
    String sql;
    Vector vCatalog = new Vector();
    try{
        sql = "SELECT * from Books WHERE category_id='"+category_ID+"' ";
        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        while(rs.next()){
            CmbBooks book = new CmbBooks();
            book.setProduct_id(rs.getString("product_id"));
            book.setTitle(rs.getString("title"));
            book.setAuthor(rs.getString("author"));
            book.setDescription(rs.getString("description"));
            book.setPrice(rs.getInt("price"));
            book.setImage(rs.getString("image"));
            vCatalog.addElement(book);
        }
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        disconnect();
    }
    return vCatalog ;
}

```

//mengambil data buku dari database berdasarkan keyword tertentu (digunakan)

```

public Vector getBooksCatalogSearch(String title){
    String sql;
    Vector vCatalog = new Vector();
    try{
        sql = "SELECT * from Books WHERE title LIKE '%" + title + "%' ";
        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        while(rs.next()){
            CmbBooks book = new CmbBooks();
            book.setProduct_id(rs.getString("product_id"));
            book.setTitle(rs.getString("title"));
            book.setAuthor(rs.getString("author"));
            book.setDescription(rs.getString("description"));
            book.setPrice(rs.getInt("price"));
            book.setImage(rs.getString("image"));
            vCatalog.addElement(book);
        }
    }
}

```

```

    }catch(Exception e){
        e.printStackTrace();
    }finally{
        disconnect();
    }
    return vCatalog ;
}

//untuk mengambil data buku yang sedang di promosikan (digunakan)
public Vector getPromotionBooks(){
    String sql;
    Vector vCatalog = new Vector();
    try{
        sql = "SELECT * from Books WHERE product_id IN (SELECT product_id
FROM promotion)" ;
        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        while(rs.next()){
            CMbBooks book = new CMbBooks();
            book.setProduct_id(rs.getString("product_id"));
            book.setTitle(rs.getString("title"));
            book.setAuthor(rs.getString("author"));
            book.setDescription(rs.getString("description"));
            book.setPrice(rs.getInt("price"));
            book.setImage(rs.getString("image"));
            vCatalog.addElement(book);

        }
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        disconnect();
    }
    return vCatalog ;
}

//mengambil data buku dari database berdasarkan product_idnya
public CMbBooks getBook(String product_ID){
    String sql;
    CMbBooks book = new CMbBooks();
    try{
        sql = "SELECT * from Books WHERE product_id= '"+product_ID+"' " ;
        connect();
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql) ;

        while(rs.next()){

            book.setProduct_id(rs.getString("product_id"));
            book.setTitle(rs.getString("title"));
            book.setAuthor(rs.getString("author"));
            book.setDescription(rs.getString("description"));
            book.setPrice(rs.getInt("price"));
            book.setImage(rs.getString("image"));

        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

```

```

        }finally{
            disconnect();
        }
        return book ;
    }
}

```

Implementasi pada logika presentasi

index2.jsp

```

<%-- --%>
<%-- lakukan import class Vector, class CmbBook, class CatalogBean dan class-
class pada package java.sql --%>

<%-- buat objek oBooks dari class CmbBooks (javabean) pada package com.jsp
gunakan scope="page"--%>
<%-- buat objek catalogBean dari class CatalogBean pada package com.jsp
(javabean) gunakan scope="page" --%>
<%-- panggil fungsi getPromotionBooks() untuk mendapatkan data buku-buku yang
sedang dipromosikan simpan dalam Vector vCatalog --%>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<body>

    <h3> Selamat Datang di <b>BookStore.com </b>
    <br>Kita menawarkan beragam buku yang menarik dengan harga special.<br>

        <%
        //lakukan iterasi
        Iterator it = vCatalog.iterator();
        while(it.hasNext()){
            //ambil object dari vCatalog simpan pada objek oBooks (gunakan method
next)
        %>
    <br> <br>

    <table>
    <tr>
        <td rowspan="7"></td>
        <td> <%-- menampilkan judul --%></b></td>
    </tr>
    <tr>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td>Author : <%-- menampilkan pengarang --%> </td>
    </tr>
    <tr>
        <td>Description : <%-- menampilkan judul deskripsi dari buku --%> </td>
    </tr>
    <tr>
        <td>Price : <%-- menampilkan harga --%> </td>

```

```

</tr>
<tr>
  <td>&nbsp;</td>
</tr>
<tr>
  <td>Quantity : <!-- buat text field dengan name amount size=3 dan value
1 dan button submit -->
  </td>
</form>
</tr>
</table>

  <%
  }
  %>
</h3>

  <!-- Buatlah sebuah form untuk pelanggan dimana pelanggan ingin mencari
buku-buku berdasarkan kata kunci yang diinginkan.
Definisi form : gunakan method post dan action="catalog.jsp?action=search"-
Terdapat sebuah textfield untuk memasukkan keyword beri nama dengan keyword dan
button submit.
%>

</body>
</html>

```

Output dari indek2.jsp

Untitled Document - Opera Opera Widgets


File Edit View Bookmarks Feeds Tools Help


New tab Untitled Document


Go to Web address here Google

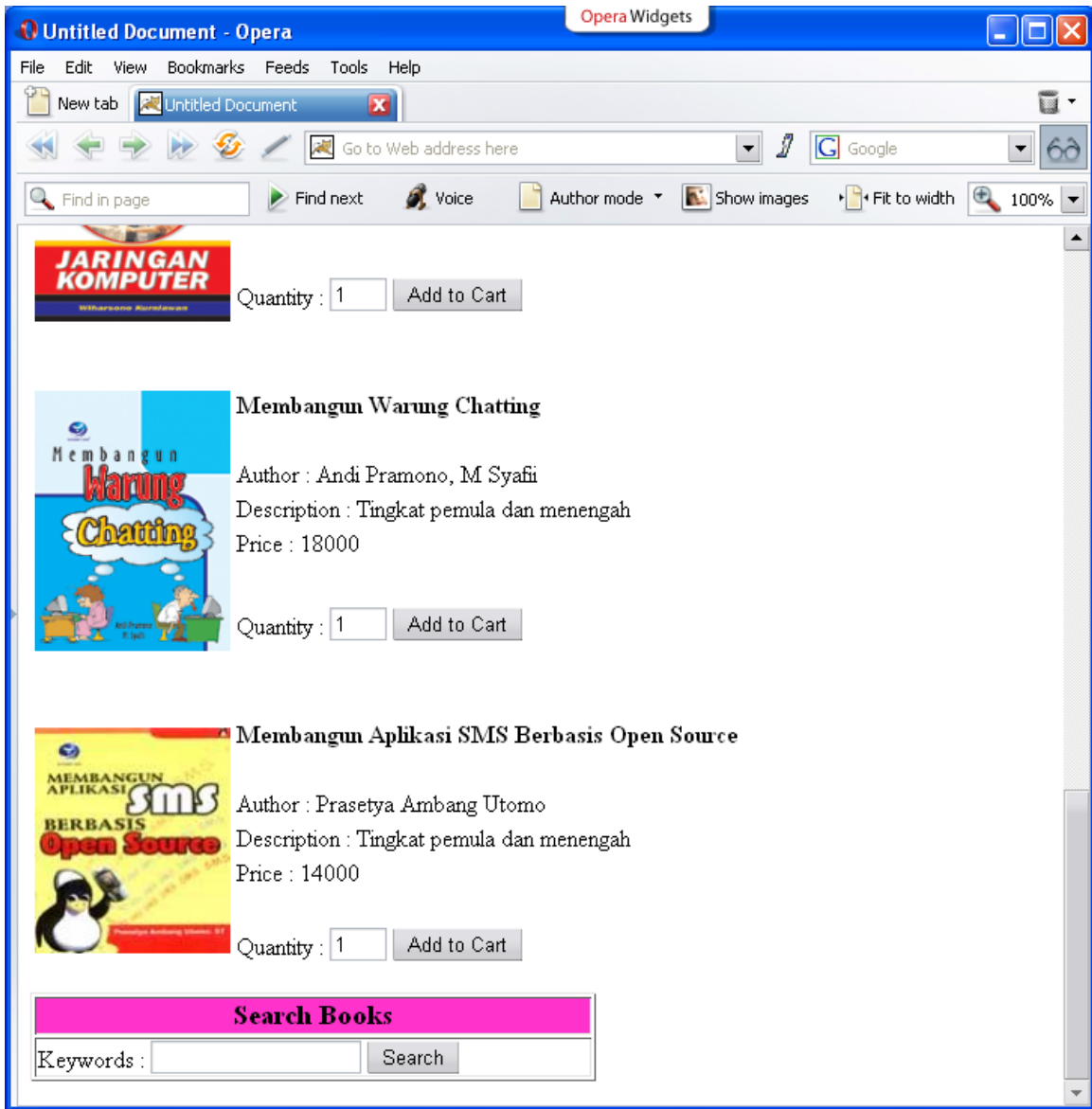
Find in page Find next Voice Author mode Show images Fit to width 100%

Selamat Datang di BookStore.com
Kita menawarkan beragam buku yang menarik dengan harga special.

 **Mengungkap Rahasia Pembuatan Virus + Worm menggunakan VBScript**
Author : Agha Abdurahman Natasyah
Description : Tingkat pemula dan menengah
Price : 35000
Quantity :

 **Pemrograman Visual Basic .NET 2005**
Author : Wahana Komputer
Description : Tingkat pemula dan menengah
Price : 58500
Quantity :

 **Panduan Praktis Pemrograman Delphi 8**



catalog.jsp

```
<!-- lakukan import class Vector, class CmbBook, class CatalogBean dan class-
class pada package java.sql -->
```

```
<!-- buat objek oBooks dari class CmbBooks (javabean) pada package com.jsp
gunakan scope="page"-->
```

```
<!-- buat objek catalogBean dari class CatalogBean pada package com.jsp
(javabean) gunakan scope="page" -->
```

```
<%
```

```
Vector vCatalog ;
String sTitle ;
String category_ID = "C01" ;
```

```
//panggil method getBooksCatalog() dengan paramter category_ID
```

```

//ambil parameter action (dengan menggunakan request.getParameter("action"))
simpan dalam variabel String sAction
String sAction = request.getParameter("action") ;
// jika sAction null maka sAction = "view"
// jika sAction sama dengan "search" maka
    // ambil parameter "keyword" simpan dalam variabel String sKeyword
    //panggil method getBooksCatalogSearch masukkan sKeyword sebagai parameter
dari fungsi tsb. Simpan hasil pada Vector vCatalog.
// jika sAction sama dengan "view" maka
    //ambil parameter id simpan dalam variabel String dengan nama sProductID
    //panggil method getBooksCatalogSearch masukan sKeyword sebagai parameter
dari fungsi tsb. Simpan hasil pada Vector vCatalog.

```

```

%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>

    <h3> Selamat Datang di <b>BookStore.com </b>
    <br>Kita menawarkan beragam buku yang menarik dengan harga special.<br>

    <%
      //lakukan iterasi
      Iterator it = vCatalog.iterator();
      while(it.hasNext()){
        //ambil object dari vCatalog simpan pada objek oBooks (gunakan method
next)
      %>
    <br> <br>

    <table>
    <tr>
      <td rowspan="7"></td>
      <td> <!-- menampilkan judul --%--></b></td>
    </tr>
    <tr>
      <td>&nbsp;</td>
    </tr>
    <tr>
      <td>Author : <!-- menampilkan pengarang --%--> </td>
    </tr>
    <tr>
      <td>Description : <!-- menampilkan judul deskripsi dari buku --%--> </td>
    </tr>
    <tr>
      <td>Price : <!-- menampilkan harga --%--> </td>
    </tr>
    <tr>
      <td>&nbsp;</td>
    </tr>
    <tr>
      <td>Quantity : <!-- buat text field dengan name amount size=3 dan value
1 dan button submit --%-->

```



```

        </td>

        </form>
    </tr>
</table>

    <%
    }
    %>
</h3>

```

`<!--` Buatlah sebuah form untuk pelanggan dimana pelanggan ingin mencari buku-buku berdasarkan kata kunci yang diinginkan.
 Definisi form : gunakan method post dan action="catalog.jsp?action=search"-
 Terdapat sebuah textfield untuk memasukkan keyword beri nama dengan keyword dan button submit.
`%>`

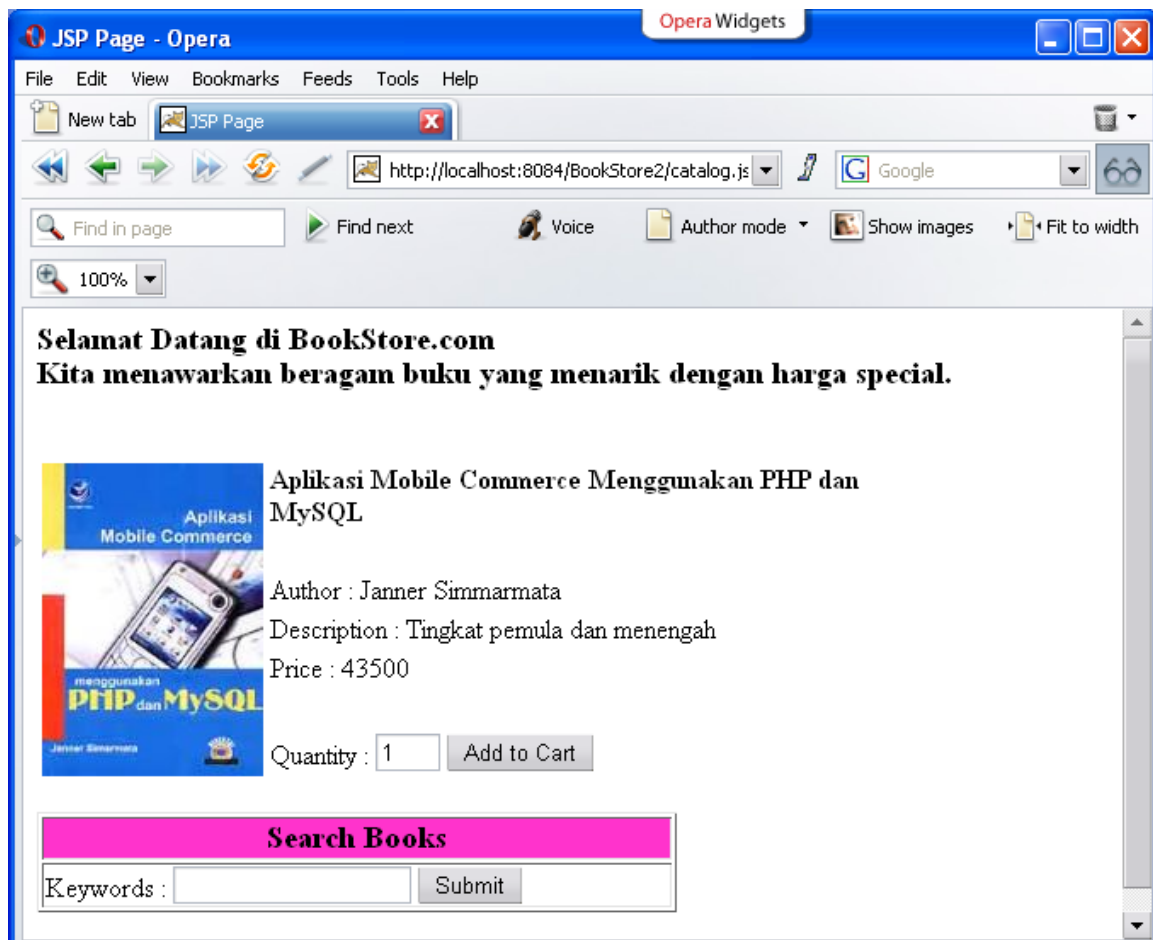
```

</body>
</html>

```

Output dari catalog.jsp

Jika user memasukkan keyword "mobile" dan selanjutnya tekan button submit maka akan menampilkan semua buku yang mengandung keyword "mobile"



index3.jsp

Untuk menampilkan semua kategori buku

```
<%-- lakukan import class Vector, class CmbBook, class CatalogBean dan class-
class pada package java.sql --%>

<%-- buat objek oBooks dari class CmbBooks (javabean) pada package com.jsp
gunakan scope="page"--%>
<%-- buat objek catalogBean dari class CatalogBean pada package com.jsp
(javabean) gunakan scope="page" --%>
<%-- panggil fungsi getAllCatalog simpan dalam variabel vector vCategory --%>

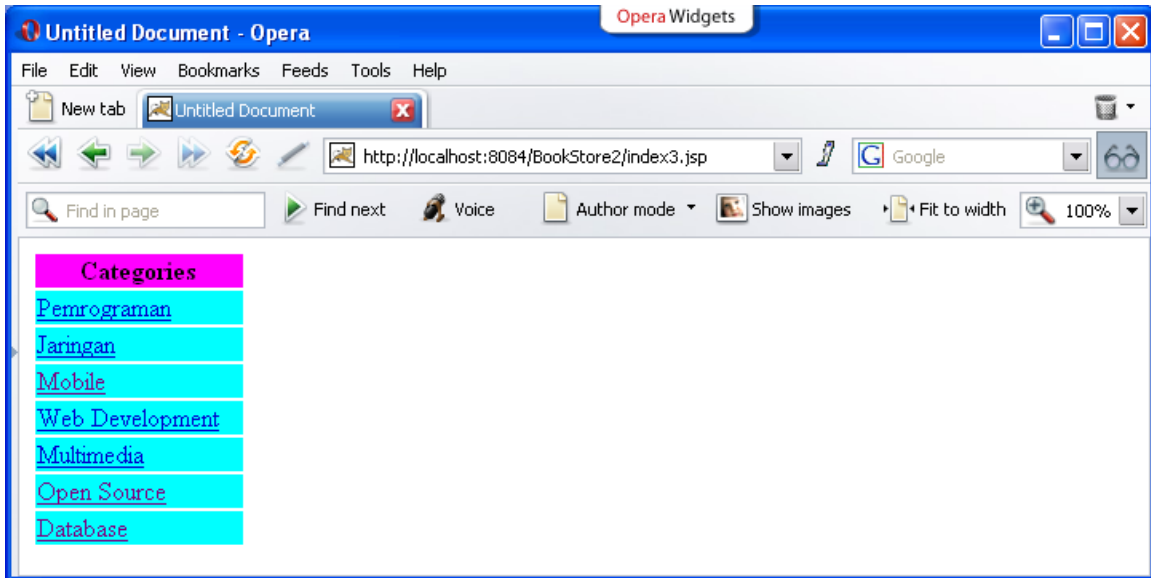
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<body>
  <table border="0">
    <tr>
      <td> Categories </td>
    </tr>
    <%
      for(int i=0;i<vCategory.size();i++){
        //ambil data pada indeks ke I simpan dalam variabel temp dengan tipe
Vector
        //ambil data pada indeks ke 0 pada objek temp simpan dalam variabel
String dengan nama id
        //ambil data pada indeks ke 1 pada objek temp simpan dalam variabel
String dengan nama nm

      %>
    <tr>
      <td><a href="catalog.jsp?action=view&id=<%-- panggil var id--%"><%--
panggil var nm--%></a></td>
    </tr>

    <%
      }
    %>
  </table>
</body>
</html>
```

Output index3.jsp



Shopping Cart

Komponen lain yang umumnya terdapat pada suatu toko online adalah keranjang belanja (shopping cart), yang memungkinkan pelanggan untuk menyimpan sementara barang yang akan dibeli pada saat memilih barang-barang lainnya. Selain itu pelanggan juga dapat mengeluarkan suatu barang dari shopping cart.

Pada aplikasi ini, komponen shopping cart dibuat dalam dua bagian yaitu `shoppingCartBean.java` dan `shopcart.java` yang membentuk logika bisnis dan logika presentasi. Shopping cart menggunakan scope session sehingga shopping cart ini akan hilang jika pelanggan keluar dari session dengan cara menutup web-browser yang digunakan.

Struktur data pada shopping cart.

Struktur data yang digunakan adalah:

1. `VectorCart` berisi `VectorContent`
2. `VectorContent` berisi:

- `CmbBooks` (data satu buku)
- `Integer amount` (jumlah buku yang akan dibeli)

Pada shopping cart, misal pelanggan membeli buku A sebanyak 2, buku B sebanyak 1, buku C sebanyak 3. Buku buku yang dibeli ini disimpan dalam `VectorCart`. `VectorCart` merupakan vector yang didalamnya terdapat `vector(VectorContent)` pula. `VectorContent` ini pada indek ke 0 menyimpan objek `Buku` dan pada indek ke 1 menyimpan jumlah buku yang akan dibeli.

VectorCart

VectorContent
Buku A (CmbBooks)
New Integer(2)

VectorContent
Buku B (CMbBooks)
New Integer(1)
VectorContent
Buku C (CMbBooks)
New Integer(3)

shoppingCartBean.java

```

package com.jsp;

import java.util.Iterator;
import java.util.Vector;

public class shoppingCartBean {
    //keranjang belanja
    private Vector vCart = new Vector();

    //untuk mengetahui total jumlah barang yang ada di shooping cart.
    private int iCartContent ;

    public Vector getCart(){
        return vCart ;
    }

    public int getCartContentNumber(){
        return iCartContent ;
    }

    //fungsi untuk menambahkan ke shopping cart
    public boolean addToCart(String product_id, int num){
        CatalogBean oBean = new CatalogBean();
        CMbBooks book = oBean.getBook(product_id);
        Vector vCartContent = new Vector();
        Integer amount = new Integer(num);
        vCartContent.addElement(book);

        vCartContent.addElement(amount);
        vCart.addElement(vCartContent);
        //update total jumlah barang yang ada di shopping cart
        iCartContent = iCartContent + amount.intValue();

        return true;
    }

    //menghapus barang dari shopping cart berdasarkan product_id
    public void removeFromCart(String product_id){
        for(int i=0;i<vCart.size();i++){
            Vector temp = (Vector)vCart.get(i);
            CMbBooks book= (CMbBooks) temp.get(0);
            Integer amount = (Integer) temp.get(1);
            String book_productid = book.getProduct_id() ;

            if (book_productid.equals(product_id)){
                vCart.removeElementAt(i);
                break;
            }
        }
    }
}

```

```

    }
}

// untuk mengecek apakah barang sudah terdapat di shopping cart, jika sudah
ada maka return true jika belum ada maka return false
public boolean isExist(String cartID){
    Iterator it = vCart.iterator() ;
    while (it.hasNext()){
        Vector vCartContent = (Vector) it.next() ;
        CmbBooks oBook = (CmbBooks) vCartContent.elementAt(0);
        String productID= oBook.getProduct_id();
        if (productID.equals(cartID))
            return true ;
    }
    return false;
}

public void destroyCart(){
    vCart = new Vector();
}
}

```

Shopcart.jsp

```

<!-- lakukan import class Vector, class CmbBook, class CatalogBean dan class-
class pada package java.sql -->

<!-- buat objek oBooks dari class CmbBooks (javabean) pada package com.jsp
gunakan scope="page"-->
<!-- buat objek catalogBean dari class CatalogBean pada package com.jsp
(javabean) gunakan scope="page" -->

```

```

<%
    Vector vShopCart = new Vector();
    String sAction;
    String sMessage = "Your Shopping Cart is Empty" ;
    int iTotal = 0 ;

    //ambil paramter action simpan dalam variabel Saction.
    // jika sAction tidak sama dengan null (add,delete) maka
    //jika sAction sama dengan "add"
        // ambil parameter "id" simpan dalam variabel String sProductID
        // ambil parameter "amount" simpan dalam variabel String sAmount
        // ubah sAmount menjadi tipe int simpan dalam variabel num
        // jika keranjang belanja belum ada buku dengan sProductID maka
        // tambahkan ke shooping cart
    // jika sAction sama dengan "delete"
        // ambil parameter "productID" simpan dalam variabel String sProductID
        // panggil method removeFromCart() dengan masukan sProductID
    // panggil method getCart() , simpan hasilnya dalam variabel vShopCart
%>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

```

```

<h3> Selamat Datang di <b>BookStore.com </b>
<br><br> Shopping Cart<br>
</h3>
<table border="1">
  <tr>
    <td>ID</td>
    <td>Title</td>
    <td>Amount</td>
    <td>Price</td>
    <td>SubTotal</td>
    <td>Delete</td>
  </tr>

  <%
    Iterator it = vShopCart.iterator();
    while(it.hasNext()){
      //ambil data untuk setiap indek pada vector vShopCart (data
dalam bentuk vector simpan dengan nama vContent. vContent ini pada indeks ke 0
berisi objek buku dan indeks ke 1 berisi jumlah dari objek buku)
      //ambil data pada indeks ke 0 simpan dalam variabel oBook
      //ambil data pada indeks ke 0 simpan dalam variabel iAmount
dengan tipe Integer
      //dapatkan harga buku tsb simpan dalam variabel iPrice (int)
      //hitung subtotal harga buku
      //hitung pula total harga buku

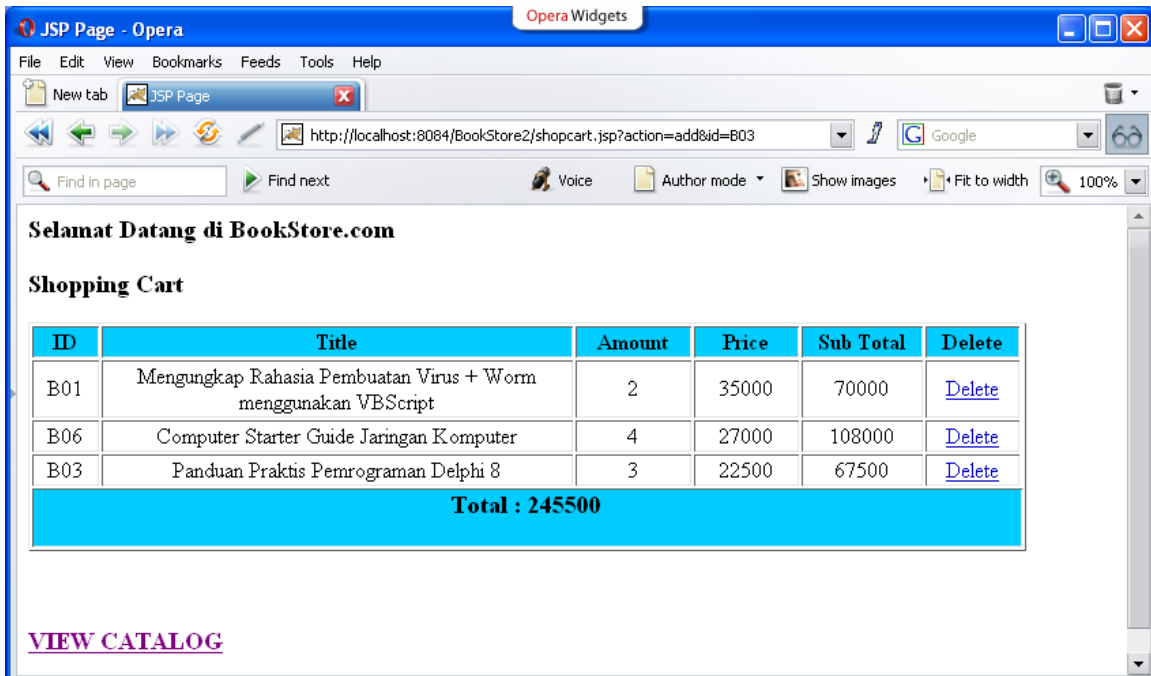
      %>

      <tr>
        <td><!-- menampilkan id Buku --></td>
        <td><!-- menampilkan judul Buku --></td>
        <td><!-- menampilkan jumlah buku --></td>
        <td><!-- menampilkan harga buku --></td>
        <td><!-- menampilkan subTotal (harga buku * jumlah buku) --></td>
        <td><a
href="shopcart.jsp?action=delete&productID=<%=oBook.getProduct_id()%>">Delete</
a></td>
      </tr>
      <%
        }
      %>
      <tr>
        <td colspan="7" ><h3>Total : <!-- menampilkan total dari semua
buku --></h3></td>
      </tr>
    </table>
    <br>
    <br>

    <a href="index2jsp"><h3>VIEW CATALOG<h3></a>
  </body>
</html>

```

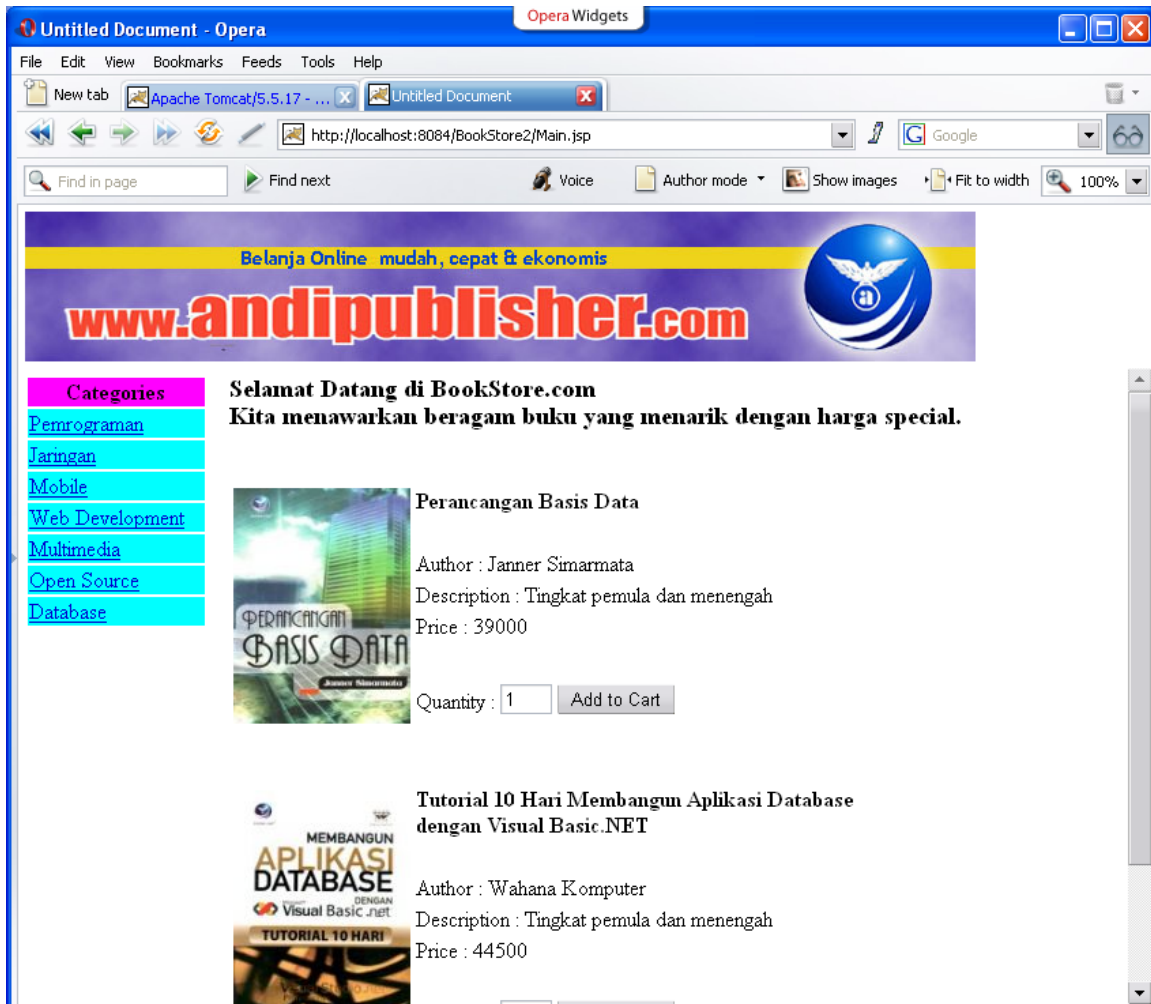
Output dari shopcart.jsp



Tampilan utama dari aplikasi ini ditunjukkan pada gambar 1. User juga dapat melihat katalog buku berdasarkan kategori buku. Misal user memilih kategori database maka hasil seperti gambar 2.



Gambar 1



Gambar 2

Referensi :

Sri Hartati Wijono, Pemrograman Java Servlet dan JSP dengan Netbeans, Penerbit Andi.
Agus Setyabudi dan Albert Samuel, Aplikasi E-Commerce dengan Java Servlet dan JSP, Penerbit Elex Media Komputindo