Videre med Java web applikationer

Denne artikel viser nogle af de mange forskellige måder man kan lave Java web applikationer på. Og giver vejledning til hvad der er godt og mindre godt.

Den forudsætter kendskab til Java og lidt JSP.

Skrevet den **16. Feb 2009** af **arne_v** I kategorien **Programmering / JSP** | ★★★☆☆

Historie:
V1.0 - 24/07/2005 - original
V1.1 - 25/07/2005 - fix typo

Indledning

Jeg vil vise 7 forskellige måder at lave præcis den samme Java web applikation på.

Eksemplet er meget banalt, men alligevel nok til at illustrere forskellene på de 7 måder.

Eksemplerne er testet med Tomcat men bør virke i enhver JSP/servlet engine.

Jeg bruger MySQL som database, men det kan rettes bare ved at rette driver navn og connection URL.

Det er ikke production quality kode. Der er ikke gjordt noget ud af:
- HTML (brug CSS til at styre det visuelle)
- sikkerhed (brug altid PreparedStatement for at undgå SQL injection)
- performance (brug altid connection pool og genbrug prepared statements etc.)
- error handling (brug et logging framework til at logge fejl og giv brugeren en pæn fejl besked)

fordi fokus er på opdelingen af HTML tags og Java kode.

[koden understøtter heller ikke URL rewriting, så sessions virker ikke uden cookies]

Jeg vil kort forklare nogle af de tekniske finesser, men en detaljeret gennemgang af de ting som bliver berørt vil kræve en meget tyk bog.

Vurderings kriterier

Hvad er en god web applikation ?

Primært: kode som er nem at læse og dermed nem at vedligeholde.

Sekundært: kode som er nem at genbruge.

[grunden til at kode genbrug kun er sekundær er at det
ofte er svært at genbruge ude i web laget]

Mange års erfaring siger at kode som er nem at vedligeholde
er kode med en skarp opdeling mellem layout (HTML tags) og
funktionalitet (Java kode).

Målsætningen er derfor:
 - ingen HTML tags i Java koden
 - ingen embedded Java kode i HTML koden

Og det vil blive brugt til at vurdere de forskellige løsninger.

Eksemplet

Eksemplet er en simpel side til at vedligeholde en kontakt database
med navn + telefon nummer + email adresse.

Funktionaliteten er ren vis og gem - ingen business logic.

Data strukturen er simpel.

SQL:

```
CREATE TABLE contacts (
    name varchar(40) NOT NULL,
    phone varchar(20),
    email varchar(40),
    PRIMARY KEY (name)
);
```

1. ren servlet

Dette var hvad man brugte førend JSP blev opfundet.

Stilen svarer meget til gode gamle CGI scripts.

dir stuktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen1>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen1

13-07-2005  23:41    <DIR>          .
13-07-2005  23:41    <DIR>          ..
13-07-2005  23:45    <DIR>          WEB-INF
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen1\WEB-INF
```

```
13-07-2005  23:45    <DIR>          .
13-07-2005  23:45    <DIR>          ..
13-07-2005  23:41    <DIR>          classes
13-07-2005  23:45               487 web.xml
               1 File(s)            487 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen1\WEB-INF\classes

13-07-2005  23:41    <DIR>          .
13-07-2005  23:41    <DIR>          ..
13-07-2005  23:46    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen1\WEB-INF\classes\test

13-07-2005  23:46    <DIR>          .
13-07-2005  23:46    <DIR>          ..
24-07-2005  11:07             3.905 ContactsServlet.class
               1 File(s)          3.905 bytes

     Total Files Listed:
               2 File(s)          4.392 bytes
              11 Dir(s)  117.062.066.176 bytes free
```

ContactsServlet.java:

```
package test;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ContactsServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        show(response.getWriter());
    }
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/test");
```

```
            Statement stmt = con.createStatement();
            String name = request.getParameter("name");
            String phone = request.getParameter("phone");
            String email = request.getParameter("email");
            stmt.executeUpdate("INSERT INTO contacts(name,phone,email) VALUES
('" + name + "','" + phone + "','" + email + "')");
            stmt.close();
            con.close();
        } catch (ClassNotFoundException e) {
            throw new ServletException(e.getMessage());
        } catch (SQLException e) {
            throw new ServletException(e.getMessage());
        }
        show(response.getWriter());
    }
    private void show(PrintWriter out) throws ServletException {
        out.println("<!doctype html public '-//w3c/dtd HTML 4.01
Transitional//en'>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Kontakter</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Kontakter</h1>");
        out.println("<h2>Eksisterende kontakter:</h2>");
        out.println("<table border>");
        out.println("<tr>");
        out.println("<th>Navn</th>");
        out.println("<th>Telefon</th>");
        out.println("<th>Email</th>");
        out.println("</tr>");
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/test");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT name,phone,email FROM
contacts");
            while(rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getString("name") + "</td>");
                out.println("<td>" + rs.getString("phone") + "</td>");
                out.println("<td>" + rs.getString("email") + "</td>");
                out.println("</tr>");
            }
            rs.close();
            stmt.close();
            con.close();
        } catch (ClassNotFoundException e) {
            throw new ServletException(e.getMessage());
        } catch (SQLException e) {
            throw new ServletException(e.getMessage());
        }
        out.println("</table>");
        out.println("<h2>Tilføj kontakter:</h2>");
```

```
        out.println("<form method='post' action='ContactsServlet'>");
        out.println("Navn: <input type='text' name='name' size='40'>");
        out.println("<br>");
        out.println("Telefon: <input type='text' name='phone' size='20'>");
        out.println("<br>");
        out.println("Email: <input type='text' name='email' size='40'>");
        out.println("<br>");
        out.println("<input type='submit' value='Tilføj'>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet>
        <servlet-name>ContactsServlet</servlet-name>
        <servlet-class>test.ContactsServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ContactsServlet</servlet-name>
        <url-pattern>/ContactsServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Start URL = http://localhost:8080/gen1/ContactsServlet

Forklaring:
 - servlet koden bør være selvforklarende, doGet kaldes ved GET og doPost kaldes ved POST
 - web.xml mapper URL /ContactsServlet -> logisk navn ContactsServlet -> servlet class test.ContactsServlet

Vurdering:
 ikke god - Java koden er fuld af HTML tags

Og ovenstående er ikke engang slemt. Jeg har undgået problemet med " i HTML tag attributter ved at bruge '. Men det går helt galt hvis man i sin Java kode skal outputte HTML med embedded JavaScript som igen outputter HTML.

Java web apps med ren servlet er stort set uddød ligesom de CGI scripts som de er modelleret efter.

Alle de out.println's gør kode ulæselig.

2. ren JSP

Man opfandt så JSP for at slippe for out.println's.

Man skriver bare HTML kode direkte og embedder Java kode i <% %>.

Helt ligesom ASP og PHP.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen2>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen2

13-07-2005  23:42    <DIR>          .
13-07-2005  23:42    <DIR>          ..
13-07-2005  23:42             1.331 contacts.jsp
13-07-2005  23:43    <DIR>          WEB-INF
               1 File(s)          1.331 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen2\WEB-INF

13-07-2005  23:43    <DIR>          .
13-07-2005  23:43    <DIR>          ..
13-07-2005  23:43               187 web.xml
               1 File(s)            187 bytes

     Total Files Listed:
               2 File(s)          1.518 bytes
               5 Dir(s)  117.062.037.504 bytes free
```

contacts.jsp:

```
<%@ page import="java.sql.*" %>
<%
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost/test");
Statement stmt = con.createStatement();
if(request.getMethod().equals("POST")) {
    String name = request.getParameter("name");
    String phone = request.getParameter("phone");
    String email = request.getParameter("email");
    stmt.executeUpdate("INSERT INTO contacts(name,phone,email) VALUES ('" +
name + "','" + phone + "','" + email + "')");
}
%>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
<head>
<title>Kontakter</title>
</head>
<body>
```

```
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<%
ResultSet rs = stmt.executeQuery("SELECT name,phone,email FROM contacts");
while(rs.next()) {
%>
<tr>
<td><%=rs.getString("name")%></td>
<td><%=rs.getString("phone")%></td>
<td><%=rs.getString("email")%></td>
</tr>
<%
}
%>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="contacts.jsp">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
</web-app>
```

Start URL = http://localhost:8080/gen2/contacts.jsp

Forklaring:
- lige ud af landevejen, alt udenfor <% %> sætter JSP compieren en out.println omkring, request er et kendt objekt i JSP ligesom response/session/application

Vurdering:
lidt bedre men stadig ikke god - vi er sluppet af med HTML tags i Java koden, men vi har til

gengæld fået embedded Java kode i HTML koden

Det ligner ASP og PHP, eller mere præcist: det ligner ASP og PHP når det er værst (ustruktureret kode som er vokset ved knopskydning skrevet af folk uden erfaring med design af større systemer - der må være en million små hjemmesider som ligner).

3. JSP med Java beans

Meget af den embedded Java kode kan relativt nemt flyttes ud af JSP siden. Det er nemlig nemt at bruge eksterne klasser i JSP.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen3>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen3

21-07-2005  22:45    <DIR>          .
21-07-2005  22:45    <DIR>          ..
21-07-2005  22:45             1.051 contacts.jsp
13-07-2005  23:51    <DIR>          WEB-INF
               1 File(s)          1.051 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen3\WEB-INF

13-07-2005  23:51    <DIR>          .
13-07-2005  23:51    <DIR>          ..
13-07-2005  23:41    <DIR>          classes
13-07-2005  23:43               187 web.xml
               1 File(s)            187 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen3\WEB-INF\classes

13-07-2005  23:41    <DIR>          .
13-07-2005  23:41    <DIR>          ..
21-07-2005  22:43    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen3\WEB-INF\classes\test

21-07-2005  22:43    <DIR>          .
21-07-2005  22:43    <DIR>          ..
24-07-2005  11:24               907 Contacts.class
24-07-2005  11:25             2.221 ContactsDatabaseAccess.class
               2 File(s)          3.128 bytes

     Total Files Listed:
               4 File(s)          4.366 bytes
              11 Dir(s)  117.061.988.352 bytes free
```

contacts.jsp:

```
<%@ page import="test.*,java.util.*" %>
<jsp:useBean id="newc" scope="request" class="test.Contacts"/>
<jsp:setProperty name="newc" property="*"/>
<%
if(request.getMethod().equals("POST")) {
    ContactsDatabaseAccess.insert(newc);
}
%>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
<head>
<title>Kontakter</title>
</head>
<body>
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<%
List all = ContactsDatabaseAccess.getAll();
for(int i = 0; i < all.size(); i++) {
Contacts c = (Contacts)all.get(i);
%>
<tr>
<td><%=c.getName()%></td>
<td><%=c.getPhone()%></td>
<td><%=c.getEmail()%></td>
</tr>
<%
}
%>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="contacts.jsp">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

Contacts.java:

```
package test;

import java.io.Serializable;

public class Contacts implements Serializable {
    private String name;
    private String phone;
    private String email;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

ContactsDatabaseAccess.java:

```
package test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class ContactsDatabaseAccess {
    public static void insert(Contacts c) throws ClassNotFoundException,
SQLException {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/test");
        Statement stmt = con.createStatement();
        stmt.executeUpdate("INSERT INTO contacts(name,phone,email) VALUES ('"
+ c.getName() + "','" + c.getPhone() + "','" + c.getEmail() + "')");
        stmt.close();
        con.close();
```

```
    }
    public static List getAll() throws ClassNotFoundException, SQLException {
        List res = new ArrayList();
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/test");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT name,phone,email FROM
contacts");
        while(rs.next()) {
            Contacts c = new Contacts();
            c.setName(rs.getString("name"));
            c.setPhone(rs.getString("phone"));
            c.setEmail(rs.getString("email"));
            res.add(c);
        }
        rs.close();
        stmt.close();
        con.close();
        return res;
    }
}
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
</web-app>
```

Start URL = http://localhost:8080/gen3/contacts.jsp

Forklaring:
- Contacts er en bean klasse
- ContactsDatabaseAccess kan indsætte en Contacts og hente en ArrayList af Contacts
- <jsp:useBean id="newc" scope="request" class="test.Contacts"/> henter et Contacts objekt fra request og hvis ikke det er der så opretter det et og gemmer i request
- <jsp:setProperty name="newc" property="*"/> henter alle form felter fra request og sætter det tilsvarende properties i objektet

Vurdering:
begynder at være rimeligt godt men der er stadig en del embedded Java kode i JSP siden og specielt den for løkke ser ikke pæn ud

4. custom taglib

JSP har imidlertid en feature til at udvide JSP sproget med så man kan lave server side logik ved hjælp af tags.

Den feature hedder taglibs.

Og vi kan erstatte den for løkke med et tag.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen4>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen4

23-07-2005  21:22    <DIR>          .
23-07-2005  21:22    <DIR>          ..
23-07-2005  21:22               952 contacts.jsp
23-07-2005  19:22               832 testtags.tld
21-07-2005  22:55    <DIR>          WEB-INF
               2 File(s)          1.784 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen4\WEB-INF

21-07-2005  22:55    <DIR>          .
21-07-2005  22:55    <DIR>          ..
13-07-2005  23:42    <DIR>          classes
13-07-2005  23:43               187 web.xml
               1 File(s)            187 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen4\WEB-INF\classes

13-07-2005  23:42    <DIR>          .
13-07-2005  23:42    <DIR>          ..
23-07-2005  19:18    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen4\WEB-INF\classes\test

23-07-2005  19:18    <DIR>          .
23-07-2005  19:18    <DIR>          ..
24-07-2005  11:24               907 Contacts.class
24-07-2005  11:25             2.221 ContactsDatabaseAccess.class
23-07-2005  19:57             2.462 List2TableRowsTag.class
               3 File(s)          5.590 bytes

     Total Files Listed:
               6 File(s)          7.561 bytes
              11 Dir(s)  117.061.976.064 bytes free
```

contacts.jsp:

```
<%@ page import="test.*" %>
<%@ taglib uri="testtags.tld" prefix="testtags" %>
<jsp:useBean id="newc" scope="request" class="test.Contacts"/>
```

```
<jsp:setProperty name="newc" property="*"/>
<%
if(request.getMethod().equals("POST")) {
    ContactsDatabaseAccess.insert(newc);
}
%>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
<head>
<title>Kontakter</title>
</head>
<body>
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<testtags:list2tablerows data="<%=ContactsDatabaseAccess.getAll()%>"
cols="name,phone,email"/>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="contacts.jsp">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

testtags.tld:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.2//EN" "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.2</jsp-version>
    <short-name>testtags</short-name>
    <tag>
        <name>list2tablerows</name>
        <tag-class>test.List2TableRowsTag</tag-class>
        <body-content>empty</body-content>
        <attribute>
            <name>data</name>
            <required>true</required>
```

```
            <rtexprvalue>true</rtexprvalue>
            <type>java.util.List</type>
        </attribute>
        <attribute>
            <name>cols</name>
            <required>true</required>
            <rtexprvalue>false</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```

List2TableRowsTag.java:

```
package test;

import java.beans.Expression;
import java.io.IOException;
import java.util.List;

import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

public class List2TableRowsTag extends TagSupport {
    private List data;

    private String cols;

    public int doStartTag() {
        String[] props = cols.split(",");
        try {
            JspWriter out = pageContext.getOut();
            for(int i = 0; i < data.size(); i++) {
                out.println("<tr>");
                for(int j = 0; j < props.length; j++) {
                    Object o;
                    try {
                        Expression expr = new Expression(data.get(i), "get" +
Character.toUpperCase(props[j].charAt(0)) +

props[j].substring(1), new Object[0]);
                        expr.execute();
                        o = expr.getValue();
                    } catch (Exception e) {
                        o = "";
                    }
                    out.print("<td>");
                    out.print(o);
                    out.print("</td>");
                    out.println();
                }
                out.println("</tr>");
            }
```

```
        } catch (IOException e) {
        }
        return SKIP_BODY;
    }

    public int doEndTag() {
        return EVAL_PAGE;
    }

    public List getData() {
        return data;
    }

    public void setData(List data) {
        this.data = data;
    }

    public String getCols() {
        return cols;
    }

    public void setCols(String cols) {
        this.cols = cols;
    }
}
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
</web-app>
```

Contacts.java og ContactsDatabaseAccess.java er de samme som tidligere.

Start URL = http://localhost:8080/gen4/contacts.jsp

Forklaring:
- <%@ taglib uri="testtags.tld" prefix="testtags" %> peger på vores taglib
- testags.tld definerer/erklærer vores tag
- klassen List2TableRowsTag implementerer vores tag
- <testtags:list2tablerows data="<%=ContactsDatabaseAccess.getAll()%>" cols="name,phone,email"/> kalder vores tag

Vurdering:
ikke så meget bedre - vi slap af med for løkken men fik til gengæld HTML tags i vores Java kode igen

Vi har brug for noget finere hvor vi har et tag for start på løkken og et tag for

slut på løkken og hvor vi så kan putte rigtige HTML tags ind imellem.

Det kan man selvfølgelig lave selv, men det er jo nemmere hvis man kan finde noget færdigt.

5. JSTL

JSTL (Java Standard Tag Library) er et færdigt tag library med tags til nogle af de mest gængse features på web sider inkl. løkker.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen5>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen5

23-07-2005  22:13    <DIR>          .
23-07-2005  22:13    <DIR>          ..
23-07-2005  22:13             1.110 contacts.jsp
23-07-2005  22:11    <DIR>          WEB-INF
               1 File(s)          1.110 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen5\WEB-INF

23-07-2005  22:11    <DIR>          .
23-07-2005  22:11    <DIR>          ..
23-07-2005  22:11    <DIR>          classes
23-07-2005  22:29    <DIR>          lib
23-07-2005  21:45               280 web.xml
               1 File(s)            280 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen5\WEB-INF\classes

23-07-2005  22:11    <DIR>          .
23-07-2005  22:11    <DIR>          ..
23-07-2005  21:23    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen5\WEB-INF\classes\test

23-07-2005  21:23    <DIR>          .
23-07-2005  21:23    <DIR>          ..
24-07-2005  11:24               907 Contacts.class
24-07-2005  11:25             2.221 ContactsDatabaseAccess.class
               2 File(s)          3.128 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen5\WEB-INF\lib

23-07-2005  22:29    <DIR>          .
23-07-2005  22:29    <DIR>          ..
28-01-2004  22:06            16.923 jstl.jar
28-01-2004  22:06           319.542 standard.jar
               2 File(s)        336.465 bytes
```

```
    Total Files Listed:
              6 File(s)        340.983 bytes
             14 Dir(s)  117.061.361.664 bytes free
```

contacts.jsp:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page import="test.*" %>
<jsp:useBean id="newc" scope="request" class="test.Contacts"/>
<jsp:setProperty name="newc" property="*"/>
<%
if(request.getMethod().equals("POST")) {
    ContactsDatabaseAccess.insert(newc);
}
request.setAttribute("allc", ContactsDatabaseAccess.getAll());
%>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
<head>
<title>Kontakter</title>
</head>
<body>
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<c:forEach var="c" items="${allc}">
<tr>
<td><c:out value="${c.name}"/></td>
<td><c:out value="${c.phone}"/></td>
<td><c:out value="${c.email}"/></td>
<tr>
</c:forEach>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="contacts.jsp">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
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" version="2.4">
</web-app>
```

Contacts.java og ContactsDatabaseAccess.java er de samme som tidligere.

Start URL = http://localhost:8080/gen5/contacts.jsp

Forklaring:
- <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> peger på JSTL core tag lib (der er andre end core)
- i web.xml har jeg skiftet fra servlet 2.3 til servlet 2.4 specifikationen for at kunne bruge JSTL og EL (Expression Language)
- c:forEach og c:out taggene må være selvforklarende

Vurdering:
nu begynder det at se godt ud - vi har kun en lille bitte smule embedded Java kode tilbage hvor vi tester for POST, laver en insert og henter data som skal vises

Hvis vi skal af med det så skal vi ikke POST'e til siden selv men POST'e til en servlet.

6. JSP + servlet

Så vi prøver at POST'e til en servlet.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen6>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen6

23-07-2005  22:31    <DIR>          .
23-07-2005  22:31    <DIR>          ..
23-07-2005  22:31               915 contacts.jsp
23-07-2005  22:39    <DIR>          WEB-INF
               1 File(s)            915 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen6\WEB-INF

23-07-2005  22:39    <DIR>          .
23-07-2005  22:39    <DIR>          ..
13-07-2005  23:42    <DIR>          classes
23-07-2005  22:29    <DIR>          lib
23-07-2005  22:39               572 web.xml
```

```
               1 File(s)            572 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen6\WEB-INF\classes

13-07-2005  23:42    <DIR>          .
13-07-2005  23:42    <DIR>          ..
23-07-2005  22:37    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen6\WEB-INF\classes\test

23-07-2005  22:37    <DIR>          .
23-07-2005  22:37    <DIR>          ..
24-07-2005  11:24               907 Contacts.class
24-07-2005  11:25             2.221 ContactsDatabaseAccess.class
24-07-2005  12:52             1.533 InsertServlet.class
               3 File(s)          4.661 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen6\WEB-INF\lib

23-07-2005  22:29    <DIR>          .
23-07-2005  22:29    <DIR>          ..
28-01-2004  22:06            16.923 jstl.jar
28-01-2004  22:06           319.542 standard.jar
               2 File(s)        336.465 bytes

     Total Files Listed:
               7 File(s)        342.613 bytes
              14 Dir(s)  117.061.292.032 bytes free
```

contacts.jsp:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page import="test.*" %>
<%
request.setAttribute("allc", ContactsDatabaseAccess.getAll());
%>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
<head>
<title>Kontakter</title>
</head>
<body>
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<c:forEach var="c" items="${allc}">
```

```
<tr>
<td><c:out value="${c.name}"/></td>
<td><c:out value="${c.phone}"/></td>
<td><c:out value="${c.email}"/></td>
<tr>
</c:forEach>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="InsertServlet">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

InsertServlet.java:

```
package test;

import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class InsertServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String name = request.getParameter("name");
        String phone = request.getParameter("phone");
        String email = request.getParameter("email");
        Contacts newc = new Contacts();
        newc.setName(name);
        newc.setPhone(phone);
        newc.setEmail(email);
        try {
            ContactsDatabaseAccess.insert(newc);
        } catch (ClassNotFoundException e) {
            throw new ServletException(e.getMessage());
        } catch (SQLException e) {
            throw new ServletException(e.getMessage());
        }
        response.sendRedirect("contacts.jsp");
    }
}
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
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" version="2.4">
    <servlet>
        <servlet-name>InsertServlet</servlet-name>
        <servlet-class>test.InsertServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>InsertServlet</servlet-name>
        <url-pattern>/InsertServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Contacts.java og ContactsDatabaseAccess.java er de samme som tidligere.

Start URL = http://localhost:8080/gen6/contacts.jsp

Forklaring:
- alt er kendte teknikker bare sat sammen på en anden måde

Vurdering:
næsten i mål - vi har kun en enkelt linie embedded Java kode tilbage
i HTML kodem hvor vi henter data der skal vises

For at slippe af med den skal vi kalde en servlet som forwarder til
JSP siden.

Men vi er ikke så glade for alt for mange servlets, så det var smart hvis
vi kunne bruge den samme servlet og bare kalde noget forskelligt Java kode
fra afhængig af kontekst.

7. Struts

Struts er et framework hvor alle requests routes til en central servlet,
kalder det rigtige Java kode og forwarder til den rigtige JSP side.

dir struktur:

```
C:\Jakarta\tomcat-5.5.9\webapps\gen7>dir /s
Volume in drive C has no label.
Volume Serial Number is E850-F261

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen7

23-07-2005  23:39    <DIR>          .
23-07-2005  23:39    <DIR>          ..
```

```
23-07-2005  23:39               810 contacts.jsp
23-07-2005  23:54    <DIR>          WEB-INF
               1 File(s)            810 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen7\WEB-INF

23-07-2005  23:54    <DIR>          .
23-07-2005  23:54    <DIR>          ..
13-07-2005  23:42    <DIR>          classes
23-07-2005  23:57    <DIR>          lib
23-07-2005  23:36               937 struts-config.xml
23-07-2005  23:54               771 web.xml
               2 File(s)          1.708 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen7\WEB-INF\classes

13-07-2005  23:42    <DIR>          .
13-07-2005  23:42    <DIR>          ..
23-07-2005  23:50    <DIR>          test
               0 File(s)              0 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen7\WEB-INF\classes\test

23-07-2005  23:50    <DIR>          .
23-07-2005  23:50    <DIR>          ..
24-07-2005  11:24               907 Contacts.class
24-07-2005  11:25             2.221 ContactsDatabaseAccess.class
24-07-2005  12:59             2.082 InsertAction.class
24-07-2005  12:59             1.535 ShowAction.class
               4 File(s)          6.745 bytes

Directory of C:\Jakarta\tomcat-5.5.9\webapps\gen7\WEB-INF\lib

23-07-2005  23:57    <DIR>          .
23-07-2005  23:57    <DIR>          ..
12-09-2004  17:35           118.726 commons-beanutils.jar
12-09-2004  17:35           175.426 commons-collections.jar
12-09-2004  17:35           109.096 commons-digester.jar
28-01-2004  22:06            16.923 jstl.jar
28-01-2004  22:06           319.542 standard.jar
12-09-2004  17:35           526.578 struts.jar
               6 File(s)      1.266.291 bytes

     Total Files Listed:
              13 File(s)      1.275.554 bytes
              14 Dir(s)  117.061.296.128 bytes free
```

contacts.jsp:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!doctype html public "-//w3c/dtd HTML 4.01 Transitional//en">
<html>
```

```
<head>
<title>Kontakter</title>
</head>
<body>
<h1>Kontakter</h1>
<h2>Eksisterende kontakter:</h2>
<table border>
<tr>
<th>Navn</th>
<th>Telefon</th>
<th>Email</th>
</tr>
<c:forEach var="c" items="${allc}">
<tr>
<td><c:out value="${c.name}"/></td>
<td><c:out value="${c.phone}"/></td>
<td><c:out value="${c.email}"/></td>
<tr>
</c:forEach>
</table>
<h2>Tilføj kontakter:</h2>
<form method="post" action="Insert.do">
Navn: <input type="text" name="name" size="40">
<br>
Telefon: <input type="text" name="phone" size="20">
<br>
Email: <input type="text" name="email" size="40">
<br>
<input type="submit" value="Tilføj">
</form>
</body>
</html>
```

ShowAction.java:

```
package test;

import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class ShowAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                 ActionForm form,
```

```
                                HttpServletRequest request,
                                HttpServletResponse response) throws
IOException, ServletException {
        try {
            request.setAttribute("allc", ContactsDatabaseAccess.getAll());
        } catch (ClassNotFoundException e) {
            throw new ServletException(e.getMessage());
        } catch (SQLException e) {
            throw new ServletException(e.getMessage());
        }
        return mapping.findForward("ok");
    }
}
```

InsertAction.java:

```
package test;

import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;

public class InsertAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response) throws
IOException, ServletException {
        try {
            DynaActionForm daf = (DynaActionForm)form;
            String name = (String)daf.get("name");
            String phone = (String)daf.get("phone");
            String email = (String)daf.get("email");
            Contacts newc = new Contacts();
            newc.setName(name);
            newc.setPhone(phone);
            newc.setEmail(email);
            ContactsDatabaseAccess.insert(newc);
            request.setAttribute("allc", ContactsDatabaseAccess.getAll());
        } catch (ClassNotFoundException e) {
            throw new ServletException(e.getMessage());
        } catch (SQLException e) {
            throw new ServletException(e.getMessage());
```

```
        }
        return mapping.findForward("ok");
    }
}
```

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
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" version="2.4">
    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
        <load-on-startup>5</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>
```

struts-config.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN"

"http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd" >
<struts-config>
    <form-beans>
        <form-bean name="insertForm"
type="org.apache.struts.action.DynaActionForm">
            <form-property name="name" type="java.lang.String"/>
            <form-property name="phone" type="java.lang.String"/>
            <form-property name="email" type="java.lang.String"/>
        </form-bean>
    </form-beans>
    <action-mappings>
        <action path="/Show" type="test.ShowAction">
            <forward name="ok" path="/contacts.jsp"/>
        </action>
        <action path="/Insert" type="test.InsertAction" name="insertForm">
            <forward name="ok" path="/contacts.jsp"/>
        </action>
```

```
    </action-mappings>
</struts-config>
```

Contacts.java og ContactsDatabaseAccess.java er de samme som tidligere.

Start URL = http://localhost:8080/gen/Show.do

Forklaring:
- alle requests som matcher *.do sendes til Struts Action Servlet
- udfra URL forwarder den så til den rigtige Action (Show.do -> ShowAction, Insert.do -> InsertAction)
- den Action gør noget og forwarder så til den rigtige JSP side (ShowAction ok -> contacts.jsp, InsertAction ok -> contacts.jsp)

Vurdering:
målet er nået der er ingen embedde Java kode i HTML koden og der er ingen HTML tags i Java koden

Bemærk at Struts kommer med sine egne taglibs som man kan bruge fremfor JSTL (af indlysende grunde da Struts er ældre end JSTL !), men jeg har altså valgt i dette eksempel kun at bruge JSTL og ren HTML.

Struts introducerer også noget helt nyt, nemlig at flowet i web applikationen er blevet konfigurerbart. Konfigurations filen styrer både hvilke actions der bliver kørt for en given URL og hvilke sider der skal vises når en action returnerer en given kode. Det kan godt virke lidt komplekst men giver faktisk en god fleksibilitet i større applikationer.

[bemærk at der ikke længere er meget sammenhæng mellem URL og sidens indhold, hvilket godt kan irritere]

Konklusion

Man kan ikke sige at en af ovenstående muligheder er ultimativt bedste. Et framework som Struts opfylder vores primære mål om at adskille HTML tags og Java kode. Men der er også en pris med den kompleksistet der følger med.

Men de er ikke alle lige gode. Nogen af dem (specielt de første) er absolut ikke gode.

Men hvilken af de andre der er bedst for dit projekt må du selv vurdere.

Der er også andre muligheder end dem der er nævnt her. Du kan f.eks. prøve at søge på JSF, WebWork og Tapestry. Java web app verdenen er karakteriseret ved at der er ufatteligt mange forskellige frameworks. Og mange af tingene kan kombineres på forskellig vis.

Men denne her artikel skulle gerne have vist dig lidt af mulighederne og introduceret dig til nogle vurderings kriterier, så du har et godt grundlag for at træffe et valg.

Buzzwords

Det er med vilje at jeg ikke har klistret MVC pattern etiketten på nogle af måderne. Det bruges ellers meget når nogen skal sælge en ny måde at lave java web apps på, men jeg synes at begrebet er blevet lidt udvandet. Og MVC i en web app er ikke det samme som MVC i en GUI app.

Nogle gange taler man også om model 1 (svarende til min #2), model 1.5 (svarende til min #3 + muligvis #4 og #5) og model 2 (svarende til min #7 + muligvis #6). De begreber synes jeg heller ikke bidrager til forståelsen.

Kommentar af simonvalter d. 25. Jul 2005 | 1

Rigtig god artikel, nok en af arnes bedste. En artikel jeg kunne have brugt for et års tid siden. Ved at vise de forskellige måder giver den et rigtig godt indblik i web applikationer som mange bøger godt kunne have brugt lidt tid på.

Som en tilføjelse til konklusionen vil jeg også anbefale at man tager et kig på JSF som er et af de mange frameworks some arne nævner. Men dette er det eneste framework der er standardiseret igennem JCP og selv om det nok ikke kommer til at overtage struts inden for det næste stykke tid så tror jeg det vil få sin plads blandt de største en dag... det er efter min mening også nemmere end struts.

Kommentar af ismar d. 25. Aug 2006 | 2

Rigtig god artikel.

Kommentar af dknsj d. 25. Jul 2005 | 3

lækker artikel!!!

Kommentar af jesper-madsen d. 24. Jul 2005 | 4

God!