Bab 11

Custom Tag

POKOK BAHASAN:

- Menggunakan Custom Tag
- Tag-tag untuk Connection
- Menggunakan tag untuk Connection
- Membuat custom tag

TUJUAN BELAJAR:

Setelah mempelajari bab ini, mahasiswa diharapkan mampu:

1. Mengetahui definisi custom tag.
2. Mengetahui cara menggunakan custom tag yang sudah ada.
3. Mengetahui cara membuat custom tag.

Kelebihan JSP yang utama diantara teknologi pemrograman web lainnya adalah kemampuan yang disediakan JSP untuk membuat tag baru yang disebut custom tag. Tag ini memiliki fungsi dan kemampuan yang didefinisikan sendiri. Telah banyak custom tag yang dibuat sehingga membentuk suatu library, sebagai contoh custom tag library yang bersifat open source dan sudah banyak digunakan adalah proyek tag library dari Jakarta Project (http://jakarta.apache.org) . Selain Jakarta Project, ada juga tag library di:

- http://www.servletsuite.com
- http://www.jsptags.com
- http://www.jspin.com

Dengan adanya Custom Tag, berbagai macam fungsi menjadi mudah digunakan. Misalnya saja kita ingin membuat custom tag yang dapat menampilkan tanggal dan waktu dalam bahasa Indonesia, maka hal ini dapat dilakukan dengan menyisipkan tag yang telah dibuat. Misalnya kode custom tag yang dapat menampilkan tanggal dan waktu dalam bahasa Indonesia seperti berikut ini.

```
<tagku:tanggal />
```

11.1 Menggunakan Custom Tag

Selanjutnya akan dibahas cara penggunaan custom tag dari Jakarta Project. Custom tag dari jakarta.apache.org yang akan digunakan adalah DBTAGS yang bisa di download di:
http://jakarta.apache.org/builds/jakarta-taglibs/releses/dbtags
dokumentasinya bisa didapat di:
http://jakarta.apache.org/taglibs/doc/dbtags-doc/intro.html

DBTAGS merupakan custom tag yang memudahkan dan mengefisienkan kode program JSP. Dengan DBTAGS, kode program akan lebih singkat dibuat dan dibaca. Tag dalam tag library ini didesain sesuai dengan spesifikasi JSP 1.2. Yang diperlukan untuk menggunakan custom tag ini adalah Servlet Container yang mendukung JSP versi 1.2. Juga dapat digunakan oleh beberapa Servlet Container untuk JSP 1.1 seperti Tomcat.

Fungsi DBTAGS adalah untuk menangani pemrograman database dalam JSP yang berkaitan erat dengan JDBC. Sebelum digunakan maka DBTAGS ini harus diinstal terlebih dahulu. Berikut ini adalah langkah-langkah mengonfigurasi aplikasi web:

1. Kopikan tag library descrriptor file (dbtags.tld) ke folder WEB-INF yang merupakan subdirektori dari aplikasi web yang akan dibangun. Anggap aplikasi web belum dibangun, maka folder ROOT bisa kita pakai:
   C:\tomcat_home\webapps\Root\WEB-INF
2. Kopikan file dbtags.jar ke folder lib dalam folder WEB-INF.

Tambahkan elemen <taglib> dalam web application deployment descriptor di dalam folder /WEB-INF, yaitu web.xml. Karena file JSP akan diletakkan dalam contex Root maka file web.xml bisa ditemukan dalam C:\tomcat_home\webapps\Root\WEB-INF\web.xml. Buka file tersebut dengan text editor dan tambahkan elemen sebagai berikut:

```
<taglib>
<taglib-uri>
    http://jakarta.apache.org/taglibs/dbtags
</taglib-uri>
<taglib-location>
    /WEB-INF/dbtags.tld
</taglib-location>
</taglib>
```

Untuk menggunakan tag ini dalam halaman JSP, tambahkan direktif berikut pada tiap halaman yang menggunakannya:

```
<%@ taglib uri=”http://jakarta.apache.org/taglibs/
application-1.0” prefix=”sql”%>
```

Perhatikan nilai “sql” pada atribut prefix dari direktif tersebut, sql disini adalah nama prefix yang ingin digunakan saat mengaplikasikan tag tersebut. Prefix ini bisa diubah sesuai keinginan.

Untuk menggunakan tag library dbtags, maka sebagai contoh akan digunakan database DataFilm yang memiliki satu buah tabel yaitu tabel FILM yang tersimpan dalan database MySql.

11.2 Tag-tag untuk Connection

- connection: untuk mendapatkan objek java.sql.Connection dari DriverManager atau DataSource
- url : mendefinisikan URL JDBC dari database
- driver : menentukan JDBC driver untuk database
- jndiName : nama dari JNDI JDBC DataSource
- userId : user id untuk database
- password : password untuk database
- closeConnection : menutup java.sql.Connection

11.3 Menggunakan Tag Connection

Tag connection terdiri atas tag url, driver serta user dan password.

```
<sql:connection id=”conn1”>
<sql:url>jdbc:mysql://localhost/test</sql:url>
<sql:driver>org.gjt.mm.mysql.Driver</sql:driver>
<sql:password>passwordnya</sql:password>
</sql:connection>
```

Atribut “id” diperlukan oleh setiap tag connection, karena diperlukan supaya tag lain dapat mengenalinya.

Tag-tag untuk Statement:

- statement : membuat dan menjalankan database query
- escape sql : escape String untuk SQL query
- query : mendeklarasikan SQL Query
- execute : menjalankan perintah SQL insert, update atau delete
- wasEmpty : menjalankan body jika tag ResultSet terakhir tidak mengembalikan baris dari database
- wasNotEmpty : menjalankan body jika tag ResultSet terakhir mengembalikan baris
- rowCount : menghasilkan jumlah baris yang didapat dari database

Langkah-langkah menggunakan DBTAG secara lengkap
Pertama-tama didefinisikan objek java.sql.Connection dan menamainya dengan atribut id.

```
<sql:connection id=”conn1”>
```

Lalu untuk mendefinisikan driver JDBC yang digunakan:

```
<sql:url>jdbc:mysql://localhost/film?user=root&password=kunci
</sql:url>
```

Untuk menentukan driver JDBC yang digunakan:

```
<sql:driver>org.gjt.mm.mysql.Driver</sql:driver>
```

Lalu menutupnya dengan tag berikut:

```
</sql:connection>
```

Kemudian dilakukan pembuatan tag statement seperti berikut:

```
<sql:statement id=”stmt1” conn=”conn1”>
```

Atribut id merupakan nama identitas dari statement dan conn diisi dengan nama id dari koneksi yang dilakukan. Setelah tag tratement, didefinisikan perintah SQL yang akan dilakukan dengan menggunakan tag queri sebagai berikut:

```
<sql:query>
   select * from film
</sql:query>
```

Oleh karena hasil informasi data dari perintah SQL select disimpan dalam obyek ResultSet, maka perlu tag resulset untuk menampilkan data yang telah diperoleh. Jika

melakukan INSERT, UPDATE, DELETE, CREATE TABLE, maka perintah SQL tersebut dapat dijalankan dengan tag:

```
<sql:execute/>
```

Jika menggunakan tag resultset, tag ini memerlukan atribut id dan penggunaannya sebagai berikut:

```
<sql:resultSet id=”rset1”>
```

Kemudian dari baris yang diperoleh, dapat ditampilkan data masing-masing kolom dengan tag getColumn. Ingat bahwa posisi kolom diawali dengan satu (1) bukannya nol (0), seperti berikut:

```
<sql:getColumn position=”1”>
```

Tag resultSet akan melakukan looping secara otomatis apabila hasil informasi dari perintah SELECT lebih dari satu baris. Apabila menggunakan tag resultSet, tag ini perlu ditutup dengan kode sebagai berikut:

```
</sql:resultSet>
```

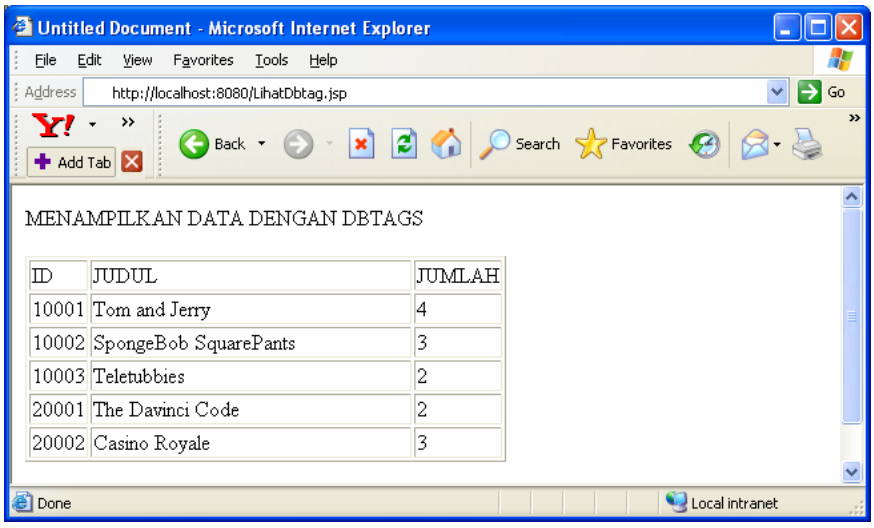
Kemudian setelah tag resultSet atau execute, tag statement dan connection perlu ditutup seperti berikut:

```
</sql:statement>
<sql:closeConnection conn=”conn1”/>
```

Penggunaan dbtag secara nyata dalam JSP untuk melihat data dengan operasi select dapat diketahui dengan mengetikkan Listing 11.1, selanjutnya simpan dengan nama LihatDbtag.jsp dan akses dengan url http://localhost:8080/LihatDbtag.jsp. Hasil ditunjukkan pada Gambar 11.1.

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/dbtags"
prefix="sql" %>

<sql:connection id="conn1">
<sql:url>jdbc:odbc:DSFilm</sql:url>
<sql:driver>sun.jdbc.odbc.JdbcOdbcDriver</sql:driver>
</sql:connection>

MENAMPILKAN DATA DENGAN DBTAGS

<table border="1">
<sql:statement id="stmt1" conn="conn1">
<sql:query>
select * from FILM
</sql:query>

<sql:resultSet id="rset1">
   <tr>
      <td><sql:getColumn position="1"/><td>
      <td><sql:getColumn position="2"/><td>
      <td><sql:getColumn position="3"/>
      <sql:wasNull>nilai null</sql:wasNull></td>
   </tr>

</sql:resultSet>
</sql:statement>
</table>


<sql:closeConnection conn="conn1"/>
```

Listing 11.1 LihatDbtag.jsp

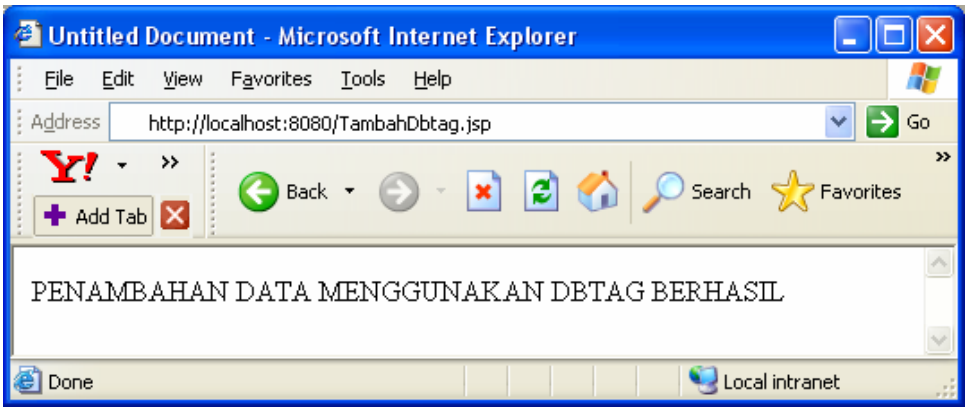


Gambar 11.1 Tampilan LihatDbtag.jsp

Untuk menambah data gunakan kode pada Listing 11.2. Hasil sebagaimana ditunjukkan pada Gambar 11.2.

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/dbtags"
prefix="sql"%>

<sql:connection id="conn1">
<sql:url>jdbc:odbc:DSFilm</sql:url>
<sql:driver>sun.jdbc.odbc.JdbcOdbcDriver</sql:driver>
</sql:connection>

<%
   String id="2003";
   String judul="Ghost Ship";
   int jumlah=5;
%>

<sql:statement id="stmt1" conn="conn1">

<%-- menset kueri sql --%>

<sql:query>
insert into FILM(id,judul,jumlah) values(
'<sql:escapeSql><%=id%></sql:escapeSql>',
'<sql:escapeSql><%=judul%></sql:escapeSql>',
'<%=jumlah%>')

</sql:query>

<%-- menjalankan kueri --%>
<sql:execute/>
</sql:statement>

<html>
<body>
PENAMBAHAN DATA MENGGUNAKAN DBTAG BERHASIL

<a href = "lihatdb.jsp"> Lihat Data </a>
</body>
</html>
```

Listing 11.2 TambahDbtag.jsp



Gambar 11.2 TampilanTambahDbtag.jsp

11.4 Percobaan

1. Buat program dengan menggunakan DBTAGS untuk menampilkan tabel FILM yang telah dibuat.
2. Buat program dengan menggunakan DBTAGS untuk menambah data pada tabel FILM yang telah dibuat.
3. Buatlah custom tag sendiri dengan langkah-langkah sebagai berikut
   a. Ketikkan Listing 12.3
   b. Simpan sebagai HelloWorldTag.java pada contex yang anda miliki dengan struktur sebagai berikut:

      C:/tomcat_home/webapps/nama_contex/
      WEB-INF/classes/tag/HelloWorldTag.java
   c. Lakukan kompilasi program HelloWorldTag.java sehingga menghasilkan file HelloWorldTag.class pada folder yang sama.
   d. Setelah dikompilasi, definisikan deskriptor untuk custom tag ini dengan membuat file yang dinamai dengan hello.tld. Kode dapat dilihat pada Listing 12.4.
   e. Tambahkan elemen untuk tag ini pada file web.xml tepatnya di

      C:/tomcat_home/webapps/nama_contex/WEB-INF/web.xml

      Sehingga menjadi seperti Listing 12.5
   f. Buat program JSP yang menggunakan custom tag Hello tersebut dengan mengetikkan program Listing 12.6.
   g. Simpan kode program JSP tersebut dengan nama hellotag.jsp.

h. Lakukan restart server Tomcat apabila sebelumnya server Tomcat telah aktif. Hal ini dilakukan supaya Tomcat dapat mengenali penambahan custom tag yang dilakukan.

i. Panggil program hellotag.jsp pada browser dengan url:

http://localhost:8080/nama_contex/hellotag.jsp

j. Browser akan menampilkan teks Hello World seperti pada Gambar 12.3.

```
package tag;

import java.io.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import javax.servlet.http.HttpServletRequest;

public class HelloWorldTag extends TagSupport{
    public int doStartTag() throws JspException{
       try{
           JspWriter out=pageContext.getOut();
           out.println("<h1>Hello World</h1>");
       }catch(IOException e){
           throw new JspException(e.toString());
       }

    }

    public int doEndTag(){
        return EVAL_PAGE;
    }
}
```

Listing 11.3 HelloWorldTag.java

```
<?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>Hello World</short-name>
  <uri>hello</uri>
```

```
   <tag>
      <name>hello</name>
      <tag-class>tag.HelloWorldTag</tag-class>
      <body-content>empty</body-content>
   </tag>
</taglib>
```

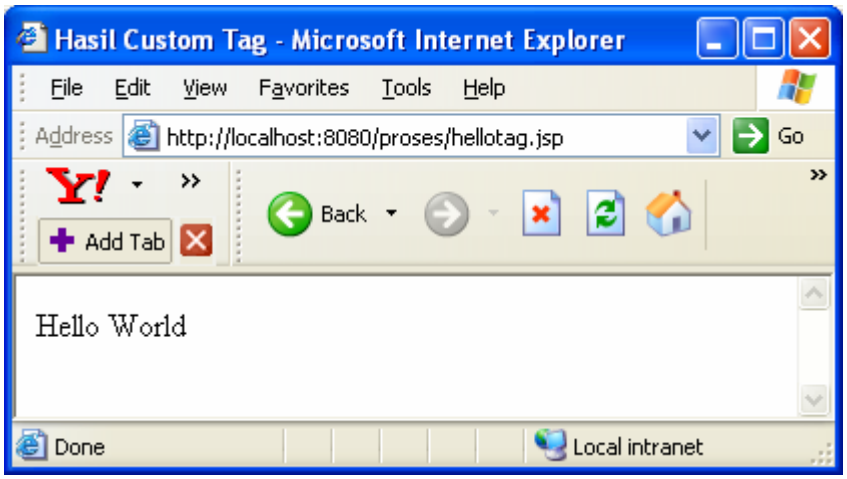
Listing 11.4 hello.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>

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

<web-app>
  <display-name>Welcome to Tomcat</display-name>
  <description>
     Welcome to Tomcat
  </description>

<taglib>
<taglib-uri>
http://jakarta.apache.org/taglibs/dbtags
</taglib-uri>
<taglib-location>
/WEB-INF/dbtags.tld
</taglib-location>
</taglib>

<taglib>
<taglib-uri>hello</taglib-uri>
<taglib-location>/WEB-INF/hello.tld</taglib-location>
</taglib>


</web-app>
```

Listing 11.5 web.xml

```
<%@ taglib uri=”hello” prefix=”util”%>
<html>
<head>
   <title>Hasil Custom Tag</title>
<head>
<body>
   <util:hello/>
</body>
</html>
```

Listing 11.6 hellotag.jsp



Gambar 11.3 Tampilan hellotag.jsp

Penjelasan :
Program HelloWorldTag.java diawali oleh package tag. Hal ini perlu dilakukan karena program java sebagai tag handler harus dikompilasi dalam folder tertentu yang didefinisikan dalam package.
Pada class ini kita sebenarnya membuat class baru yang berasal dari class TagSupport sehingga untuk mendeklarasikan class ini kita menambahkan extends TagSupport. TagSupport adalah class yang mengimplementasikan interface Tag dan masih ada class lain yang dapat digunakan, misalnya BodyTagSupport. Oleh karena CustomTag yang kita buat adalah tag tunggal dan memproses body, maka menggunakan class TagSupport.

Ada dua metode utama dari tag handler yang harus ada, yaitu:
`public int doStartTag()`
Metode ini dipanggil saat Custom Tag dijalankan.

`public int doEndTag()`
Metode ini dipanggil pada akhir Custom Tag.

Kemudian kita mendapati kode:
`return SKIP_BODY;`
Dan `return EVAL_PAGE;`

Kode SKIP_BODY berarti pada CustomTag tidak terdapat elemen dalam body dan hanya merupakan tag tunggal. Walau ada elemen dalam body, karena adanya kode ini maka elemen tersebut tidak akan diproses. Kode EVAL_PAGE memiliki arti bahwa kode JSP setelah Custom Tag akan diproses.

11.5 Soal Latihan

1. Apa yang dimaksud dengan custom tag?

2. Apa keuntungan menggunakan custom tag?
3. Bagaimana langkah-langkah menggunakan custom tag?
4. Apa tujuan mengedit file deskriptor dalam penggunaan custom tag?
5. Buatlah sebuah custom tag untuk menampilkan tanggal dan waktu saat ini!