

BAB 8

Advanced MVC

8.1 Pendahuluan

Pada pembahasan sebelumnya, kita telah memahami garis besar dasar dari Struts. Kita telah mempelajari bagaimana cara mengimplementasikan framework Struts pada aplikasi dengan mengkonfigurasi ActionServlet yang disediakan untuk menangani request. Kita juga telah mempelajari cara untuk membuat instance dari Action classes yang berfungsi sebagai action handlers pada penyerahan form dan user request yang lain. Telah diketahui cara pembuatan ActionForm classes yang menyediakan cara mudah transfer data dari form menuju ActionHandlers yang dibuat. Terakhir, telah diketahui pula mengenai penggunaan tag library untuk menggabungkan form HTML dalam halaman JSP pada framework.

Pada bab ini, kita akan membahas beberapa teknik dan fitur terapan dari Struts framework. Pertama, akan kita pelajari mengenai penggunaan DynaActionForms untuk mengurangi jumlah class yang kita gunakan dalam framework. Kemudian, akan kita telusuri bagaimana framework validator berfungsi dalam hal validasi baik pada server-side maupun client-side. Terakhir, akan dikenalkan Tiles framework, yang memiliki fitur presentationlayer yang lebih kompleks, sehingga dapat membuat templating machine terhadap halaman – halaman yang ada pada aplikasi.

8.2 DynaActionForms

Pada aplikasi skala besar utamanya, jumlah class yang perlu untuk dibuat dan dipergunakan dapat begitu tinggi. Struts mendukung classes yang cukup membantu hal ini, terutama dalam hubungannya dengan ActionForms, yang mempersyaratkan solidnya implementasi form keseluruhan pada aplikasi. Banyak dari developer yang terjebak pada batasan ini, utamanya pada saat ActionForms adalah berupa JavaBeans sederhana yang memiliki method get dan set pada tiap form fields pada representasinya.

Struts memiliki penyelesaian atas permasalahan tersebut pada versi 1.1, disebut dengan DynaActionForms. DynaActionForms sangat mirip dengan ActionForms, sehingga sebuah instance darinya dapat diambil dan method dipanggil pada saat ActionHandlers memerlukan data yang ada. Perbedaan utamanya adalah, tiap DynaActionForm tidak terdeklarasi sebagai class terpisah, melainkan terkonfigurasi dalam struts-config.xml.

Dibawah ini adalah contoh konfigurasi dan deklarasi DynaActionForms. Kita gunakan sesuai contoh pada bab sebelumnya, disini dibuat sebuah ActionForm yang akan menangani data yang dibutuhkan pada saat log in.

```
<?xml version="1.0"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.1//EN" "http://jakarta.apache.org/struts/dtds/struts-
config_1_1.dtd" >
<struts-config>
  <form-beans>
    <form-bean
      name="loginForm"
      type="org.apache.struts.action.DynaActionForm">
      <form-property name="loginName" type="java.lang.String"/>
      <form-property name="password" type="java.lang.String"/>
    </form-bean>
  </form-beans>
  <action-mappings>
    <action name="loginForm"
      path="/login"
      scope="request"
      type="login.LoginAction">
      <forward name="success" path="/success.jsp"/>
      <forward name="failure" path="/failure.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

Seperti yang terlihat, pembuatan DynaActionForm berlangsung dengan amat sederhana, dan pada beberapa kasus, deklarasi DynaActionForm dalam file konfigurasi lebih sederhana dan cepat dibandingkan dengan menuliskan instance ActionForm secara langsung. Kita tidak perlu memperhatikan form properties serta membuat method get dan set pada setiap ActionForm, dengan menggunakan DynaActionForm hanya memerlukan deklarasi property name dan type.

Berikut ini adalah Java types yang didukung oleh DynaActionForm :

- o java.lang.BigDecimal
- o java.lang.BigInteger
- o boolean dan java.lang.Boolean
- o char dan java.lang.Character
- o double dan java.lang.Double
- o float dan java.lang.Float
- o int dan java.lang.Integer
- o long dan java.lang.Long
- o short dan java.lang.Short
- o java.lang.String
- o java.lang.Date
- o java.lant.Time
- o java.sql.TimeStamp

8.3 Validators

Validasi input sangat dibutuhkan dalam aplikasi berbasis web. Dengan menggunakan validasi input, didapatkan koreksi atas format dan isi dari input yang diberikan user. Dalam banyak kasus, seorang user tidak selalu memasukkan input yang benar, huruf dapat dimasukkan dalam field numerik dan sebaliknya; sebuah field yang mempersyaratkan minimal 3 digit hanya diisi dengan 2 digit, dan sebagainya. Adalah tanggung jawab dari aplikasi untuk waspada dan menangani kesalahan input disamping kesalahan hasil dari proses business logic (password tidak cocok dan sebagainya).

Struts meringankan kesulitan yang dihadapi developer dalam menjalankan validasi dengan menyediakan validasi framework yang disebut dengan Validator Framework. Penggunaan framework ini memiliki beberapa kelebihan :

- **Framework menyediakan beberapa aturan validasi.** Terdapat beberapa rangkaian pemeriksaan umum pada aplikasi seperti pemeriksaan format, panjang karakter dan sebagainya. Framework ini menyediakan komponen – komponen yang dibutuhkan sehingga developer tidak lagi membuat kode yang akan melakukan proses validasi. Komponen yang disediakan umumnya cukup mewakili proses validasi global, namun proses validasi lain juga dapat dibuat.
- **Mengurangi redudansi kode.** Framework memisahkan komponen yang melakukan validasi dengan komponen yang memerlukan validasi. Hal ini juga membuat developer dalam penggunaan ulang kode dengan lebih mudah : komponen yang memerlukan validasi cukup mendeklarasikan jenis validasi yang diinginkan disamping menambahkan kode validasi pada komponen tersebut.
- **Satu titik maintenance.** Developer tidak lagi menelusuri aplikasi secara menyeluruh untuk memeriksa aturan validasi yang terdapat pada bermacam komponen. Seluruh aturan dideklarasikan pada file konfigurasi yang diberikan oleh framework.

Terdapat beberapa langkah yang dibutuhkan dalam menambahkan fungsi – fungsi validator dalam aplikasi Struts :

- Konfigurasi Validator Plug-in
- Deklarasi form yang memerlukan validasi, dan tipe validasi yang dibutuhkan
- Membuat pesan yang akan ditampilkan jika proses validasi mengalami kegagalan
- Merubah base class dari ActionForms menjadi ValidatorForm pada aplikasi. Dan juga merubah tipe dari DynaActionForms menjadi DynaValidatorForm

8.3.1 Konfigurasi Validator Plug-in

Langkah ini diperlukan untuk membuat Struts framework mengetahui penggunaan Validator framework. Hal yang perlu dilakukan adalah menambahkan beberapa baris kode dalam struts-config.xml. Berikut ini contohnya :

```
...
        <forward name="success" path="/success.jsp"/>
        <forward name="failure" path="/failure.jsp"/>
        </action>
    </action-mappings>

    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property
            property="pathnames"
            value="/WEB-INF/validator-rules.xml, /WEB-INF/validation.xml"/>
    </plug-in>

</struts-config>
```

Path names property menginformasikan pada framework dimana letak untuk menemukan file konfigurasi yang diperlukan. Terdapat beberapa file konfigurasi : validator-rules.xml dan validator.xml.

8.3.2 validator-rules.xml

File konfigurasi ini mendefinisikan classes yang mengimplementasi kode validasi. Framework menyertakan file ini dengan validator classes yang telah didefinisikan sebelumnya.

Dibawah ini adalah daftar logical names dari validators yang disertakan dalam framework :

- o **required** – properties yang diberi tanda sebagai required harus memiliki minimal satu karakter disamping spasi
- o **mask** – properties dengan mask validator harus sesuai dengan format penulisan yang dicantumkan pada validator dengan menggunakan **mask** variable
- o **minlength** – property tersebut harus memiliki panjang minimal sama atau lebih dari **min** variable yang diberikan
- o **maxlength** – property tersebut harus memiliki panjang yang sama atau kurang dari **max** variable yang diberikan
- o **range** – property memiliki panjang dalam area antara max dan min value yang diberikan
- o **byte, short, integer, long, float, double** – property harus sesuai dengan tipe primitive yang diberikan
- o **date** – validasi sukses jika value dari property berupa tanggal yang valid
- o **creditCard** – validasi berhasil bila value dari property adalah format nomor kartu kredit yang valid

- o **email** – validasi berhasil jika value dari property tersebut format email yang benar

8.3.3 validation-xml

File konfigurasi ini menginformasikan pada framework form apa saja yang membutuhkan validasi dan aturan validasi apa yang akan diimplementasikan. Framework ini hanya menyediakan struktur dari file. Developer harus mengkonfigurasi sendiri file ini untuk membantu fungsionalitas framework.

8.3.3.1 Menkonfigurasi file validation.xml

Sekali lagi, kita akan mempelajari elemen yang dibutuhkan oleh file konfigurasi pada contoh. Dibawah ini adalah potongan yang akan terlihat dari file jika kita akan mengkonfigurasi form login terlebih dahulu.

```
<formset>
  <form name="loginForm">
    <field property="loginName" depends="required">
      <arg0 key="error.loginname.required"/>
    </field>

    <field property="password" depends="required,minlength"/>
      <arg0 key="error.password.required"/>
      <var>
        <var-name>min</var-name>
        <var-value>4</var-value>
      </var>
    </field>
  </form>
</formset>
```

Berikut adalah macam elemen yang terdapat pada file XML ini.

<formset>

Elemen ini berfungsi sebagai container untuk elemen – elemen pada form

<form>

Setiap form dalam aplikasi yang memerlukan proses validasi harus memiliki <form> elemen yang berhubungan. Atribut name pada hal ini harus sesuai dengan name dari form-bean yang dikonfigurasi pada struts-config.xml. Elemen ini dapat mengandung satu atau lebih elemen <field>.

<field>

Tiap elemen field merepresentasikan sebuah property pada form yang memerlukan validasi. Elemen ini memiliki atribut – atribut sebagai berikut :

- o property – nama dari property dalam form yang akan divalidasi
- o depends – daftar validator yang dipisahkan dengan tanda koma yang kemudaian akan diaplikasikan pada property. Nama dari validator didefinisikan dalam validator-rules.xml.

<arg0>

Mendefinisikan key pada error message yang akan ditampilkan jika property gagal dalam proses validasi. Key dan error message yang akan ditampilkan dapat ditemukan pada resource bundle yang dibuat oleh developer.

<var>

Tiap elemen var mendefinisikan sebuah variabel dengan value masing – masing yang akan diteruskan menuju validator. Nama dari variabel didefinisikan dengan elemen turunan <var-name>, sedangkan isi value-nya didefinisikan menggunakan elemen turunan <var-value>.

Telah dijelaskan di atas informasi – informasi tentang bermacam elemen pada file konfigurasi, dapat dikatakan bahwa contoh tersebut mengkonfigurasi form login aplikasi Struts sehingga loginName dan password field harus memiliki value yang dimasukkan. Sebagai tambahan, password field harus memiliki panjang minimal 4 karakter untuk melewati proses validasi.

8.3.3.2 Mendefinisikan resource bundle

Elemen <arg0> di atas mendefinisikan sebuah key yang perlu dicocokkan dengan sebuah entry pada resource bundle. Entry tersebut kemudian digunakan untuk menentukan pesan yang akan ditampilkan pada user. Validator framework menggunakan resource bundle yang juga digunakan oleh Struts framework, dimana pada kondisi default, dapat ditemukan pada direktori WEB-INF/classes dengan nama ApplicationResources.properties. Jika file resource bundle tersebut tidak ditemukan pada aplikasi, maka buatlah sebuah text file dengan nama yang sama.

Entry pada resource bundle adalah pasangan sederhana berupa key=value. Pada konfigurasi file diatas, kita dapatkan isi file sebagai berikut :

```
error.loginname.required=Masukkan nama anda
error.password.required=Anda belum mengisi password atau password kurang dari
4 karakter
```

Langkah terakhir dalam menggunakan validation framework adalah mengubah ActionForm dan DynaActionForm classes untuk memakai classes yang disediakan. Gunakan DynaActionForm pada contoh sebelumnya :

```
<?xml version="1.0"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration
1.1//EN" "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd" >
  <struts-config>
    <form-beans>
      <form-bean
        name="loginForm"
        type="org.apache.struts.validator.DynaValidatorForm">
        <form-property name="loginName" type="java.lang.String"/>
        <form-property name="password" type="java.lang.String"/>
      </form-bean>
    </form-beans>
    <action-mappings>
      <action name="loginForm"
        path="/login"
        scope="request"
        type="login.LoginAction"
        validation="true">
        <forward name="success" path="/success.jsp"/>
        <forward name="failure" path="/failure.jsp"/>
      </action>
    </action-mappings>
  </struts-config>
```

8.4 Tiles

Framework lain yang bekerja dengan Struts adalah Tiles framework. Dengan menggunakan Tiles, kita dapat mendefinisikan templates dan screen instances yang dapat digunakan dalam aplikasi dengan mudah.

Apakah templates itu?

Secara ringkas, sebuah halaman template adalah desain layout halaman yang dapat digunakan kembali oleh halaman apapun pada aplikasi. Penggunaan templates membuat aplikasi terlihat lebih konsisten.

Untuk lebih memahami konsep templates, mari kita perhatikan beberapa halaman dari aplikasi web :

HALAMAN DARI SITUS TERPERCAYA

Seperti terlihat pada halaman – halaman tersebut, terdapat beberapa elemen desain yang dipakai. Secara keseluruhan menggunakan rangkaian navigational links yang sama pada bagian kiri dan rangkaian gambar pada bagian atas, serta bagian yang berubah antara halaman yang berbeda adalah bagian tengah halaman.

Pertimbangkan bagaimana cara kita untuk mengimplementasikan beberapa halaman yang sama. Jika kita implementasikan pada setiap halaman seperti yang terlihat, dalam hal ini tepat satu halaman yang berhubungan dengan halaman-halaman yang terlihat pada gambar, pada nantinya kita akan mengalami kesulitan pada saat melakukan proses maintenance. Hal itu dikarenakan oleh banyaknya duplikasi kode yang akan digunakan, jika pada suatu saat seseorang memutuskan bahwa gambar pada halaman tidak terlihat bagus, ataupun navigational menu pada bagian kiri tidak terlalu menarik, perubahan akan dilakukan terhadap seluruh halaman yang ada.

Seorang programmer mengetahui bahwa pembagian area kode dapat diduplikasikan pada seluruh halaman aplikasi. Konsep ini juga dapat diterapkan pada halaman JSP/HTML : navigational links dapat berupa satu halaman tersendiri, header images sebagai halaman lain, begitu pula dengan footer. Halaman – halaman ini kemudian dapat digunakan pada halaman dengan isi body. Hal ini dapat dilakukan tanpa menggunakan framework apapun, cukup menggunakan <jsp:include> action yang telah dibahas sebelumnya pada bagian dasar JSP.

Terdapat penyelesaian yang lebih baik, yaitu dengan membuat bagian body sebagai halaman bagian JSP tersendiri, kemudian membuat sebuah halaman JSP lain yang akan berfungsi sebagai default format dan layout dari halaman halaman yang ada. Seorang developer kemudian akan mengimplementasikannya dengan cara memasukkan halaman yang berisi body content dalam template tersebut.

Terlihat jelas kelebihan dari cara diatas : segala keperluan perubahan hanya akan dilakukan pada satu aspek pada presentation layer, sebagai contoh header, akan diterapkan pada seluruh halaman aplikasi hanya dengan memodifikasi satu halaman. Jika anda ingin mengganti layout dari aplikasi, disamping mengerjakan content dari aplikasi, cukup dilakukan dengan mengubah halaman template yang digunakan sebelumnya.

Berikutnya akan dibahas mengenai implementasi template menggunakan Tiles Framework :

8.4.1 Mempersiapkan Tiles

Sebelum menggunakan Tiles framework, perlu dilakukan beberapa langkah persiapan sebagai berikut :

1. Menambahkan beberapa baris kode berikut pada struts-config.xml sebelum tag penutup </struts-config>

```
<plug-in className="org.apache.struts.tiles.TilesPlugin" >
  <set-property
    property="definitions-config"
    value="/WEB-INF/tiles-defs.xml" />
</plug-in>
```

Kode ini menginformasikan pada Struts bahwa aplikasi akan menggunakan TilesFramework dan file konfigurasinya diletakkan pada direktori /WEB-INF dengan nama tiles-defs.xml

- Langkah berikutnya, salin file struts-tiles.tld dan tiles-config.dtd dari direktori library Struts menuju direktori /WEB-INF. File pertama berisi macam – macam tags yang dibutuhkan dalam penggunaan Tiles, sedangkan file kedua berisi struktur file konfigurasi dari tiles-defs.
- Buat sebuah file konfigurasi kosong yang akan diisi kemudian. Buat sebuah file XML pada direktori /WEB-INF. kemudian beri nama dengan tiles-defs.xml. Isi file tersebut dengan kode sebagai berikut :

```
<!DOCTYPE tiles-definitions PUBLIC
    "-//Apache Software Foundation//DTD Tiles Configuration//EN"
    "http://jakarta.apache.org/struts/dtds/tiles-config.dtd">

<tiles-definitions>

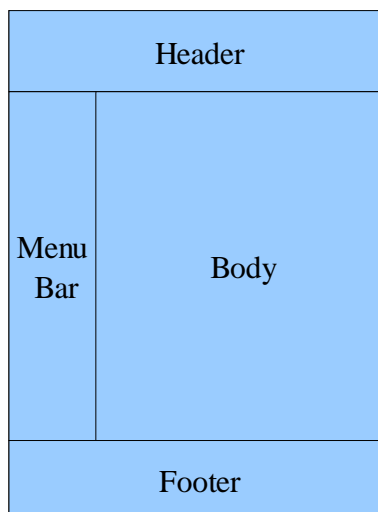
</tiles-definitions>
```

Jika langkah – langkah tersebut telah dijalankan seluruhnya, maka Tiles Framework siap digunakan.

8.4.2 Membuat Layout Template

Langkah pertama dalam pembuatan sebuah template adalah mengidentifikasi komponen yang akan ditempatkan. Sebagai contoh, sebagai besar halaman web memiliki komponen header, footer, menu bar dan body.

Berikut ini rancang diagram dasar template yang akan dibuat :



Untuk membuat sebuah halaman template yang menerapkan layout tersebut, ikuti langkah – langkah berikut ini :

1. Buat sebuah halaman JSP baru
2. Import tag library dari Tiles
3. Buat file HTML yang menerapkan layout tersebut, biarkan kosong pada bagian elemen yang digunakan
4. Gunakan tag **<tiles:insert>** yang berfungsi sebagai penentu letak dari komponen layout

Halaman HTML yang menerapkan layout tersebut memiliki bentuk sesederhana tabel, dengan komponen sebagai elemen tabel seperti terlihat pada halaman JSP di bawah ini :

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles"
%>
<HTML>
<HEAD>
    <TITLE><tiles:getAsString name="title"/></TITLE>
</HEAD>
<BODY>
<TABLE border="0" width="100%" cellspacing="5">

<TR>
    <TD colspan="2"><tiles:insert attribute="header"/></TD>
</TR>
<TR>
    <TD width="140" valign="top">
        <tiles:insert attribute="menu"/>
    </TD>
    <TD valign="top" align="left">
        <tiles:insert attribute="body"/>
    </TD>
</TR>
<TR>
    <TD colspan="2">
        <tiles:insert attribute="footer" />
    </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Terdapat dua macam tag pada contoh yang diberikan diatas : **<tiles:insert>** dan **<tiles:getAsString>**.

- o **<tiles:insert>** - tag ini cukup sering digunakan pada Tiles Framework. Dalam hal ini berfungsi untuk memasukkan bagian JSP yang direferensikan dengan nama yang terisi pada atribut **attribute**.
- o **<tiles:getAsString>** - tag ini berfungsi menampung value yang diisikan pada komponen dengan atribut **name** sebagai String.

8.4.3 Membuat Screen Definitions

Setelah memuat sebuah template, kita dapat menggunakannya untuk mendefinisikan sebuah screen. Pembuatan screen definition dapat dilakukan dalam dua cara dalam Tiles framework : didefinisikan dalam halaman JSP, atau dalam file XML yang dikenali oleh framework.

Membuat sebuah definition menggunakan halaman JSP :

Pembuatan definition dalam halaman JSP menerapkan banyak macam tag yang disertakan oleh Tiles Framework :

- o `<tiles:definition>` - tag dasar yang digunakan untuk mendefinisikan layer. Value dari attribute `id` adalah nama yang terhubung dengan komponen lain. Value dari `page` attribute adalah lokasi dimana template file tersebut berasal.
- o `<tiles:put>` - tag ini digunakan untuk menempatkan sebuah value ke dalam sebuah attribute dalam template. **Name** attribute adalah nama dari lokasi target dalam template, sedangkan **value** mendefinisikan lokasi dari fragmen JSP yang akan digunakan.

Perhatikan contoh dibawah ini :

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles"
%>
<tiles:definition id="welcomePage" page="/layout/basicLayout.jsp">
  <tiles:put name="title" value="Welcome!" />
  <tiles:put name="header" value="/header.jsp" />
  <tiles:put name="footer" value="/footer.jsp" />
  <tiles:put name="menu" value="/menu.jsp" />
  <tiles:put name="body" value="/welcome.jsp" />
</tiles:definition>
```

Pada contoh tersebut, kita membuat sebuah screen definition untuk halaman utama, yaitu halaman muka aplikasi. Halaman tersebut menggunakan layout yang didefinisikan pada daftar, dan menempatkan komponen title, header, footer, menu dan body pada lokasi yang ditentukan pada template.

Secara default, definition ini hanya terlihat pada halaman JSP yang memiliki deklarasi definition terkait. Untuk mengubahnya, kita dapat menambahkan sebuah value kedalam atribut opsional **scope** pada tag `<tiles:definition>`. Values pada atribut tersebut mencakup : `page`, `request`, `session` dan `application`.

8.4.4 Membuat definition menggunakan konfigurasi file XML

Cara kedua untuk membuat screen definitions adalah mencantumkan konfigurasi pada file tiles-defs.xml. Syntax penulisan konfigurasi tersebut sangat mirip dengan tag `<tiles:definition>` yang digunakan sebelumnya :

```
<!DOCTYPE tiles-definitions PUBLIC
    "-//Apache Software Foundation//DTD Tiles Configuration 1.1//EN"
    "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">
<tiles-definitions>
  <definition name="welcomePage" page="/layout/basicLayout.jsp">
    <put name="title" value="Welcome!"/>
    <put name="header" value="/header.jsp"/>
    <put name="footer" value="/footer.jsp"/>
    <put name="menu" value="/menu.jsp"/>
    <put name="body" value="/welcome.jsp"/>
  </definition>
  <!-- ... more definitions ... -->
</tiles-definitions>
```

Apakah perbedaan antara dua cara tersebut? Kedua cara menyimpan definition sebagai JavaBean ke dalam memory. Method pada JSP me-*load* definition setelah fragmen JSP yang berisi definition tersebut telah diakses, dan secara default, membuatnya terlihat hanya pada halaman yang sama. Kondisi ini membuat penggunaan ulang dari screen definition pada aplikasi sedikit lebih sulit, dibutuhkan perhatian ekstra untuk menghindari loading dan loading definition yang tidak penting terhadap memory.

Dilain pihak, XML method membuat screen definition dapat digunakan oleh keseluruhan aplikasi secara langsung setelah startup aplikasi. Tiles framework membuatnya permanen dalam memory. Satu-satunya hal yang diperlukan oleh komponen untuk menggunakannya adalah dengan memasukkan nama definition tersebut.

Kelebihan lain dari penggunaan XML method dalam pembuatan screen definition adalah definition itu sendiri dapat digunakan sebagai ActionForwards oleh Struts framework. Hal ini dapat mengurangi secara drastic halaman JSP yang dibutuhkan oleh aplikasi. Menggunakan definition sebagai ActionForwards merupakan pokok pembahasan yang akan dibahas lebih detail selanjutnya.

8.4.5 Menggunakan Screen Definitions

Membuat sebuah screen definition belum cukup untuk ditampilkan pada user. Supaya sebuah definition dapat dipakai, gunakan tag `<tiles:insert>` dan isikan nama definition yang akan ditampilkan :

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles" %>
<tiles:insert beanName="welcomePage" flush="true"/>
```

Salah satu permasalahan dengan pendekatan ini adalah meningkatkan jumlah halaman JSP yang dibutuhkan untuk menampilkan layer yang berbeda pada user, disamping komponen body content, tiap screen membutuhkan halaman terpisah yang akan menggunakan screen definition. Untuk sebuah aplikasi yang membutuhkan 100 screen atau lebih, jumlah halaman yang dibutuhkan akan berlipat ganda.

Pendekatan yang lebih baik adalah dengan menggunakan definition sebagai target dari `ActionForward`. Dengan menambahkan Tiles framework sebagai plugin pada Struts, Struts mengetahui screen definitions yang dibuat dengan menggunakan Tiles. Screen names berfungsi sebagai penentu lokasi pada tag `<forward>`. Perhatikan contoh di bawah ini :

```
<action-mappings>
  <action name="loginForm"
    path="/login"
    scope="request"
    type="login.LoginAction"
    validation="true">
    <forward name="success" path="/welcomePage"/>
    <forward name="failure" path="/failure.jsp"/>
  </action>
</action-mappings>
```

Disini kita melakukan modifikasi atas contoh dari `struts-config.xml` sebelumnya, sehingga pada situasi dengan kondisi **success** dipetakan pada definition **welcomePage**. Seperti yang terlihat, pendekatan ini cukup sederhana, mudah digunakan, dan mengurangi jumlah halaman JSP yang harus dibuat.

8.4.6 Menambahkan Definitions

Salah satu kekuatan dari Tiles framework dari templating method lain adalah mengijinkan extensions pada screen definitions. Screen extensions bekerja dengan cara yang sama dengan inheritance class pada Java : extended screen menurunkan seluruh properties dan attributes dari parent Definition. Hal ini mengijinkan kita untuk membuat sebuah definition dasar yang mendeklarasikan values untuk attribut yang diextend oleh definitions terutama pada halaman – halaman tertentu.

Perhatikan contoh dibawah ini :

```
<definition name="basicDefinition" page="/layout/basicLayout.jsp">
  <put name="header" value="/header.jsp"/>
  <put name="footer" value="/footer.jsp"/>
  <put name="menu" value="/menu.jsp"/>
</definition>

<definition name="welcomePage" extends="basicDefinition">
  <put name="title" value="Welcome!"/>
  <put name="body" value="/welcome.jsp"/>
</definition>

<definition name="otherPage" extends="basicDefinition">
<put name="title" value="My title"/>
<put name="body" value="/otherContent.jsp"/>
</definition>
```

Dengan pembuatan sebuah base screen yang kemudian dapat di extend, kita menghindari penulisan ulang pada definisi value attribute. Dan juga, jika diperlukan perubahan dalam sebuah komponen yang terdapat pada base attributes, perubahan tersebut dapat dengan cepat diterapkan pada seluruh screen secara otomatis dengan memodifikasi base screen definitions.