BAB 9

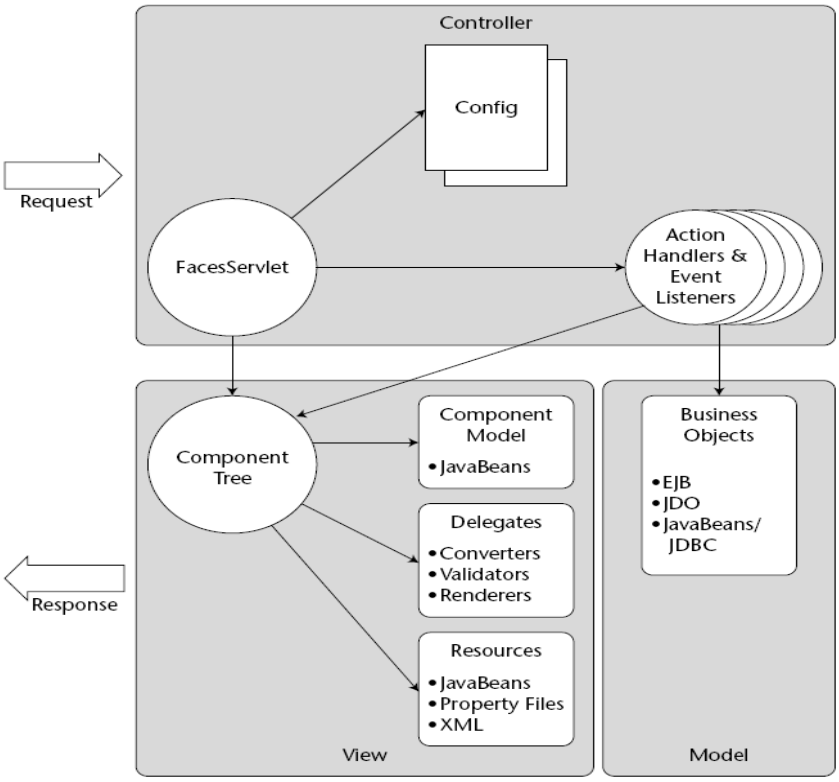
JSF

Pada bab sebelumnya, kita telah melihat Struts, sebuah framework open-source untuk aplikasi web yang mengimplementasikan arsitektur model-2. Sekarang mari kita lihat framework lainnya : Java Server Faces (JSF).

9.1 Pengenalan JSF

JSF adalah framework untuk membangun user interface untuk aplikasi web. Dibangun berdasar pada konsep-konsep yang diperkenalkan oleh Struts dan membagi bersama keuntungan sebuah arsitektur yang benar-benar memisahkan presentasi layer dari business logic dan sebuah standard komponen user interface yang perangkatnya serupa dengan widget Swing.

Dibawah ini adalah gambaran detail bagaimana framework Faces bekerja.



Gambar 9-1 JavaServerFaces Framework
(Gambar dari Mastering JavaServer Faces, Wiley Publishing)

Seperti yang kita lihat, JSF juga mempunyai pemisahan yang jelas antara komponen layer Model, View , dan Controller. Sama seperti Struts, JSF memiliki sebuah controller servlet bagian depan yaitu FacesServlet yang bertanggung jawab untuk menerima permintaan dari client dan kemudian menjalankan action yang dibutuhkan yang dituntun oleh framework. Persamaan lainnya adalah mereka berdua menggunakan action handler yang terpisah dari controller servlet bagian depan. Meskipun demikian handle-handle Faces ini sedikit berbeda dengan Struts.

Faces dan Struts memiliki tujuan yang sama berkaitan dengan layer View. Disini, Struts hanya menyediakan sebuah set libray-libray tag yang ditambahkan pada bagian atas fungsi HTML standard. Sebaliknya, Faces menyediakan set sendiri dari komponen-komponen beserta sebuah set library-library untuk memperlihatkan komponen-komponen ini sebagai tag-tag dan sebuah komponen hasil render yang menterjemahkan komponen UI menjadi HTML.

9.1.1 CONTROLLER

Layer controller dari Faces yang terdiri dari controller servlet (FacesServlet), satu set file konfigurasi XML dan sebuah set action handler.

9.1.1.1 FacesServlet

FacesServlet bertanggung jawab untuk menerima permintaan dari client dan menjalankan operasi yang diperlukan untuk menghasilkan respon. Operasi ini termasuk menyiapkan komponen-komponen UI yang dibutuhkan untuk permintaan, meng-update status komponen, memanggil action handler yang dibutuhkan (jika ada), dan komponen-komponen UI hasil render yang merupakan bagian dari respon.

FacesServlet disediakan untuk kita oleh framework JSF, dan hanya membutuhkan konfigurasi dalam sebuah pengembangan descriptor aplikasi sebelum siap untuk digunakan.

Dibawah ini adalah potongan yang ditunjukkan untuk Kita bagaimana mengkonfigurasikan FacesServlet untuk aplikasi Kita.

```
...

<servlet>
   <servlet-name>FacesServlet</servlet-name>
   <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   <load-on-startup>1</load-on-startup>
</servlet>
...

<servlet-mapping>
   <servlet-name>FacesServlet</servlet-name>
   <url-pattern>*.jsf</url-pattern>
</servlet-mapping>

...
```

9.1.1.2 Action Handlers

Telah dijelaskan sebelumnya bahwa Faces menggunakan action handlor yang independen dari controller servlet bagian depan, sama seperti Struts. Meskipun demikian, Faces mengerjakan fungsi ini dengan cara yang berbeda.

Dalam Faces, ada dua cara membuat action handler. Yang pertama adalah dengan menjadikan satu sebuah method JavaBean untuk bertindak sebagai action handler, dan yang kedua adalah dengan membuat sebuah class instans yang mengimplementasikan interface ActionListener.

9.1.1.3 *method aplikasi*

Sebuah method yang harus mempunyai suatu komponen UI untuk bertindak sebagai action handler disebut method aplikasi. Kemudian, dalam bagian View, Kita akan melihat bagaimana pengikatan selesai. Sementara itu, ada beberapa aturan yang dibutuhkan untuk membuat sebuah method aplikasi :

- Method harus dideklarasikan public
- Method harus tanpa parameter
- jenis return method harus menjadi sebuah String

Dibawah ini adalah sebuah method yang mungkin dapat kita gunakan untuk menangani sebuah event sebagai hasil dari user yang mencoba untuk login:

```
...
   public String performLogin() {
      // melarikan user ke case failure jika loginName nya kosong
      if (loginName == null || loginName.length() < 1) {
          return "failure"
      }

       // melarikan user ke case failure jika password nya kosong
       if (password == null || password.length() < 1) {
          return "failure";
       }

       User user = null;
```

```
        // membuat object business yang akan menangani pengecekan hak akses
        UserService service = new UserService();

        user = service.login(loginName, password);

        // membimbing user ke case failure jika user tidak dikenal
        if (user == null) {
            return "failure";
        }

        // mendapatkan kembali instance object FacesContext
        //  yang akan memberikan konteks yang lain kepada kita

        FacesContext ctx = FacesContext.getCurrentInstance();

        // meletakkan hasil kedalam kontek session untuk digunakan oleh komponen lain

        Map sessionMap = ctx.getExternalContext().getSessionMap();
        sessionMap.put(ApplicationConstants.USER_OBJECT, user);

        // sejak user berhasil login, melanjutkan ke success
        return "success";
    }
}
...
```

Salah satu keuntungan dari jenis action handling ini adalah mengurangi banyaknya object yang pengembang perlukan untuk di-maintain. Method ini dapat di dalam setiap JavaBean yang dikenal oleh framework, meskipun biasanya dapat ditemukan di dalam bean yang digunakan sebagai *backing model* untuk halaman form tertentu. Pada contoh diatas, *loginName* dan *password* adalah properties di dalam *backing model*.

String yang dikembalikan oleh method aplikasi menginformasikan FacesServlet dimana berikutnya dilihat oleh user. String bersifat nama-nama logis, kadang disebut *outcome*; outcome ini dicocokkan dengan peraturan-peraturan navigasi yang dituliskan dalam file konfigurasi.

Satu hal yang penting untuk dicatat adalah cara yang ditempuh oleh Kita dalam menempatkan object kedalam scope session dalam framework JSF. Dalam Struts, karena penanganan object Action form itu diberi sebuah instans HttpServletRequest, menjadi mudah untuk menerima sebuah salinan dari scope session yang mewakili object HttpSession. Bagaimanapun juga, bukan masalah untuk action handler dalam Faces. Sebagai gantinya, object di dalam framework Faces mendapat keuntungan akses untuk context external (web container, portlets, dsb) menggunakan sebuah instans dari object FacesContext.

```
        FacesContext ctx = FacesContext.getCurrentInstance();

        ...

        Map sessionMap = ctx.getExternalContext().getSessionMap();
        sessionMap.put(ApplicationConstants.USER_OBJECT, user);
```

Setelah menerima object FacesContext, Kita menerima perwakilan map object di dalam scope session dengan memanggil **getExternalContext().getSessionMap().**

9.1.1.4 *ActionListener*

Cara lain mengimplementasikan action handler dalam JSF adalah membuat sebuah class yang mengimplementasikan interface ActionListener. Interface ini menjabarkan sebuah method single :

public void processAction(ActionEvent event)

Object ActionEvent dilewatkan sebagai parameter dalam method yang menyediakan implementasi akses class kepada komponen yang disebabkan oleh event. Hal ini serupa dengan bagaimana cara object Event bekerja di dalam pemrograman Swing.

Dibawah ini adalah sebuah contoh implementasi ActionListener yang digunakan untuk logging aksi user.

```
public class PerformedActionListener implements ActionListener {
    public void processAction(ActionEvent event) {

        // mendapatkan kembali komponen yang dipakai oleh event
        UICommand component = (UICommand)event.getComponent();

        //  mendapatkan kembali nama dari button atau link
        String commandName = (String)component.getValue();

        // membuat object business yang menampilkan fungsionalitasnya
        LoggingService service = new LoggingService();

        //  menampilkan operasi logging
        service.logUserAction(commandName);
    }
}
```

Hingga saat ini, lebih baik menggunakan method aplikasi untuk bertindak sebagai action handler. Pertama, mereka dapat ditempatkan di dalam class yang sama dimana bertindak sebagai *backing model* sebuah form, dan hal seperti itu mempunyai akses lebih mudah kepada pengguna menyediakan data. Kedua, menjadi *backing model* mengijinkan pengembang untuk menggolongkan bersama-sama sebuah data dan method yang bekerja pada satu class, membuat lebih buat di-maintain. Ketiga, method aplikasi mampu mengembalikan outcome yang menginformasikan FacesServlet tampilan yang akan ditampilkan selanjutnya. ActionListener tidak bisa, dengan demikian hanya dapat membawa user kembali kepada halaman asli setelah penanganan sebuah event.

ActionListener adalah pilihan yang sesuai, meskipun demikian jika Anda ingin kemampuan refactor umum bahwa anda dapat menggunakan kembali across multiple action sources.

Jika kita lihat yang mendatang, dimungkinkan untuk memiliki sebuah method aplikasi dengan sebuah atau lebih ActionListener untuk bekerja sebagai handler untuk action tertentu. Hal terbaik untuk dilakukan kemudian adalah untuk mendapatkan terbaik dari keduanya : memiliki sebuah method aplikasi untuk mengerjakan penanganan khusus kepada action, dan kemudian memiliki ActionListener yang mengerjakan fungsi tujuan umum.

9.1.1.5 faces-config.xml

Hal ini bertindak sebagai file konfigurasi utama untuk layer controller dari framework JSF. Sebagai lawan rekan pendampingnya di dalam framework Struts, Dia tidak berisi masukan-masukan konfigurasi untuk aturan-aturan navigasi, seperti juga untuk JavaBean yang akan dikenali oleh framework.

Dibawah ini adalah contoh lengkap dari file konfigurasi, untuk sebuah aplikasi angan-angan yang mengimplementasikan kasus penggunaan login.

```
<!DOCTYPE faces-config PUBLIC
"-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
"http://java.sun.com/dtd/web-facesconfig_1_0.dtd">

<faces-config>

   <managed-bean>
      <description>
         This bean serves as the backing model for our login form
      </description>
      <managed-bean-name>loginPage</managed-bean-name>
      <managed-bean-class>sample.LoginPageBean</managed-bean-class>
      <managed-bean-scope>request</managed-bean-scope>
      <managed-property>
         <description>
            The property that will store the user's login name
         </description>
         <property-name>loginName</property-name>
         <null-value/>
       </managed-property>
       <managed-property>
         <description>
            The property that will store the user's password
         </description>
         <property-name>password</property-name>
         <null-value/>
       </managed-property>
   </managed-bean>

   <navigation-rule>
       <from-view-id>/login.jsf</from-view-id>
       <navigation-case>
           <description>
                Any failure result should bring the user to an error page
           </description>
           <from-outcome>failure</from-outcome>
           <to-view-id>/error.jsp</to-view-id>
       </navigation-case>
       <navigation-case>
           <description>
                Successful login should bring user to welcome page
           </description>
           <from-outcome>success</from-outcome>
           <to-view-id>/welcome.jsp</to-view-id>
       </navigation-case>
   </navigation-rule>

</faces-config>
```

Mari kita bahas satu per satu elemen di atas.

<!DOCTYPE ...
Digunakan untuk menunjukkan bahwa file ini bertindak sebagai sebuah file konfigurasi untuk JSF. Kegagalan untuk memasukkan baris ini, atau salah mengetik akan mengakibatkan sebuah error.

<faces-config>
Bertindak sebagai elemen utama dari file XML ini. Semua elemen lainnya harus menjadi turunan dari elemen ini.

<managed-bean>
Setiap elemen managed-bean berfungsi untuk menggambarkan suatu JavaBean yang akan diatur dan dikenal oleh framework. Dimana memiliki elemen-elemen turunan seperti berikut ini :

- <description> - tidak melayani tujuan apapun kecuali untuk memperbaiki kemampuan baca (readability) file konfigurasi.
- <managed-bean-name> - bertindak sebagai logical name dimana sebuah instans bean ini yang dapay diakses atau digunakan di dalam framework. Harus unik untuk setiap bean.
- <managed-bean-class> - nama class yang sepenuhnya berkualitas dari JavaBean untuk diatur.
- <managed-bean-scope> - scope dimana bean ini disimpan. Dapat `request`, `session`, `application`, or `none`. Memiliki sebuah nilai none berarti status bean tidak akan disimpan di tengah permintaan atau di dalam permintaan.
- <managed-property> - mendeklarasikan nilai untuk digunakan untuk inisialisasi properties di dalam JavaBean. Tidak perlu membuat masukan <managed-property> untuk setiap property di dalam JavaBean, hal tersebut hanya jika Anda ingin initialize. Memiliki elemen turunan seperti berikut :
  - <property-name> - property JavaBean untuk diatur.
  - <property-class> - (optional) menjelaskan jenis yang sesuai dari property. (tambahan)
  - <null-value/> - mengatur nilai property menjadi null.
  - <value> - memberi nilai sebuah property.

<navigation-rule>
Elemen ini digunakan untuk menjelaskan mapping logical untuk titik balik di dalam aplikasi Anda. Dapat digunakan yang mana menjelaskan aturan khusus untuk halaman tertentu, atau untuk menjelaskan aturan yang dapat digunakan oleh tiap-tiap halaman di dalam aplikasi. Memiliki elemen-elemen turunan dibawah ini :

- <from-view-id> - menggambarkan halaman yang berisi pohon komponen dimana aturan akan diterapkan. Jika tidak dikhususkan, maka aturan tersebut akan diterapkan untuk semuanya di dalam aplikasi. (tambahan)
- <navigation-case> - menjelaskan satu outcome untuk aturan navigasi. Memiliki elemen turunan seperti berikut :
  - <from-outcome> - menjelaskan outcome yang dikembalikan dari sebuah action yang akan menentukan jika navigation case menjadi aktif.
  - <to-view-id> - menjelaskan halaman berikutnya yang akan menjadi aktif jika navigation case ini menjadi aktif

Ada elemen-elemen lainnya yang tersedia untuk file konfigurasi JSF. Lihat dokumentasi implementasi JSF Anda untuk detail lebih lanjut.

Kesimpulan untuk layer Controller :

Pengaturan satu kali :

- Konfigurasi FacesServlet untuk digunakan dalam aplikasi Anda.

Untuk setiap halaman web yang berisi komponen UI JSF :

- Buatlah sebuah masukan konfigurasi untuk backing model halaman.
- Buatlah aturan navigasi dimana menjelaskan dimana aplikasi bisa mungkin pergi berikutnya setelah halaman.

9.1.2 MODEL

Framework JSF tidak menetapkan kembali setiap class atau komponen yang pengembang diwajibkan untuk terbiasa dengan menggambarkan tiap class yang mengimplementasikan business logic dari aplikasi. Bagaimanapun juga, dalam Faces dibutuhkan untuk memiliki class-class yang akan menyimpan status komponen UI dalam setiap halaman. Class-class ini disebut *backing model* elemen-elemen itu.

Class-class ini bukanlah class-class Model ketika dipandang dengan seksama dibawah perspektif dari arsitektur model 2. Bagaimanapun juga, ketika hanya berpikir komponen-komponen UI, hal tersebut bisa dipahami untuk memanggil class-class ini adalah bagian dari Model, khususnya jika Kita untuk membandingkannya dengan implementasi MVC dari class-class komponen Swing UI. Ingat, dalam Swing bahwa rendering layer disebut View, status komponen adalah Model dan action handler adalah bagian Controller.

Meskipun mereka disebut bagian dari Model, beberapa hal perlu diperhatikan yang harus diambil dalam pengembangan dari class-class ini seperti mereka tidak mempengaruhi fungsi inti dari aplikasi Anda(Model sesungguhnya). Sangat baik untuk mengingat bahwa komponen-komponen ini dimaksud untuk menyimpan status untuk komponen UI dan mungkin menjelaskan operasi dasar yang mengakses data yang tersimpan dimana dapat bertindak sebagai method aplikasi. Mereka tidak dimaksudkan untuk mengerjakan proses yang berat atau tiap-tiap proses sama sekali yang dapat digunakan kembali dalam aplikasi lain.

Membuat sebuah backing model untuk halaman yang berisi komponen UI JSF sangat mudah. Sama mudahnya seperti membuat sebuah JavaBean dengan properties yang berhubungan dengan tiap komponen dalam sebuah halaman. Dalam hal ini, mereka serupa dengan object ActionForm dalam framework Struts, dengan perkecualian bahwa mereka tidak perlu untuk meng-extend setiap class dasar yang disediakan oleh framework.

Dibawah ini adalah sebuah contoh backing model untuk sebuah form login.

```
public class LoginPageBean {
   private String loginName;
   private String password;

   public String getLoginName() {
      return loginName;
   }

   public void setLoginName(String loginName) {
      this.loginName = loginName;
   }

    public String getPassword() {
       return password;
    }

    public void setPassword(String password) {
       this.password = password;
    }
}
```

Model ini kemudian dapat diakses dengan halaman Kita setelah dikonfigurasikan dengan baik dalam file konfigurasi faces-config.xml. Konfigurasi masukan untuk bean ini adalah ditiru dibawah sebagai acuan.

```
<managed-bean>
      <description>
         This bean serves as the backing model for our login form
      </description>
      <managed-bean-name>loginPage</managed-bean-name>
      <managed-bean-class>sample.LoginPageBean</managed-bean-class>
      <managed-bean-scope>request</managed-bean-scope>
      <managed-property>
          <description>
              The property that will store the user's login name
          </description>
          <property-name>loginName</property-name>
          <null-value/>
       </managed-property>
       <managed-property>
          <description>
              The property that will store the user's password
          </description>
          <property-name>password</property-name>
          <null-value/>
       </managed-property>
   </managed-bean>
```

9.1.3 VIEW

Tak dapat dipungkiri,view adalah layer dimana kebanyakan JSF menjadi terkenal. Dalam layer ini, JSF tidak hanya menyediakan untuk Kita dengan tag-tag custom dimana kita dapat menggunakan untuk menampilkan interface Kita menggunakan JSP, Dan juga menyediakan untuk Kita dengan sebuah set komponen dan sebuah standardisasi API untuk mengakses dan memanipulasi mereka.

Sebuah diskusi komponen JSF bisa sangat diperumit jika Kita mencoba untuk menjelaskan segalanya pada waktu yang sama. Sebagai gantinya, Kita akan mengambil sebuah tampilan yang sederhana terlebih dahulu, dan menjelajah lebih dalam kemudian.

9.1.3.1 Integrasi JSF-JSP

Pertama-tama, mari mulai dengan bagian yang paling mencolok dari menggunakan komponen JSF : Bagaimana caranya menampilkan mereka kepada user. Berkut adalah listing untuk halaman JSP yang berisi komponen JSF dimana bertindak sebagai halaman login untuk contoh Kita sebelumnya.

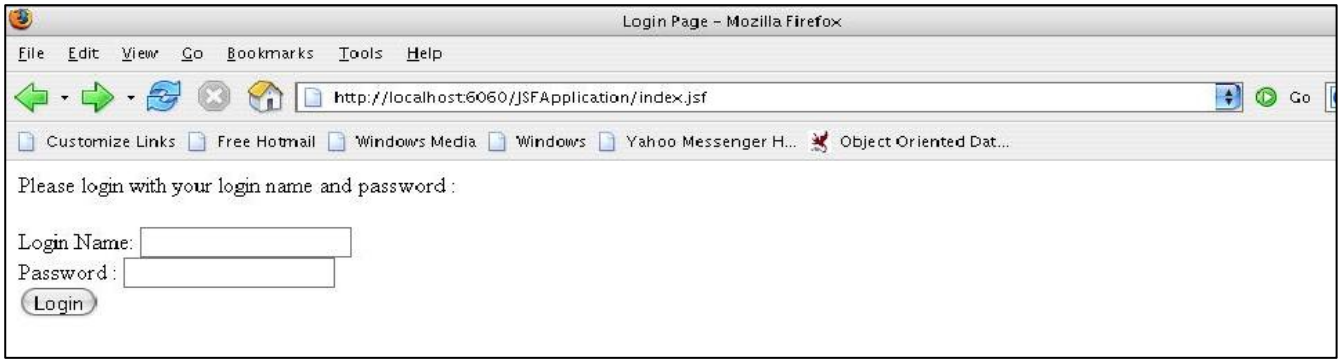
```
<%@taglib uri ="http://java.sun.com/jsf/core/" prefix="f" %>
<%@taglib uri ="http://java.sun.com/jsf/html" prefix="h" %>

<HTML>
    <TITLE>Login Page</TITLE>
    <BODY>

    Please login with your login name and password : <br/><br/>

   <f:view>
   <h:form id="simpleForm">

        <h:outputLabel for="loginName">
            <h:outputText value="Login Name:"/>
        </h:outputLabel>
        <h:inputText id="loginName" value="#{loginPage.loginName}"/><br/>

        <h:outputLabel for="password">
             <h:outputText value="Password : "/>
        </h:outputLabel>
        <h:inputSecret id="password" value="#{loginPage.password}"/><br/>

        <h:commandButton action="#{loginPage.performLogin}" value="Login"/>
    </h:form>
    </f:view>

    </BODY>
</HTML>
```

Untuk menggunakan komponen JSF dalam halaman JSP Kita, Kita butuh untuk memasukkan dua tag library : core dan html. Core tag library menjelaskan fungsi inti, seperti bagaimana cara untuk mengatur komponen JSF seperti kemampuan mereka untuk menyimpan status, dan lain-lain. HTML library menjelaskan tag-tag yang mengatakan kepada browser bagaimana cara render komponen JSF Kita menjadi sesuai dengan HTML mereka.

Begitu Kita telah memasukkan kedua tag library ini, Kita menggunakan tag-tag custom yang mereka jelaskan. Mari Kita periksa tag-tag yang telah Kita gunakan di dalam contoh-contoh yang terdahulu :

- <view> - digambarkan di dalam library inti. Semua penjelasan tag-tag komponen JSF harus enlosed didalam tag ini. Tag ini menyediakan sebuah tempat untuk implementasi JSF untuk mampu menyimpan status dari komponen UI Kita. Ingat, tidak ada view, tidak ada status yang disimpan untuk komponen-komponen Kita.
- <form> - digambarkan di dalam library HTML. Render sebuah form dalam HTML.
- <outputLabel> - menggambarkan sebuah komponen label yang berhubungan dengan komponen JSF lainnya. Komponen yang dihubungkan ditandai oleh nilai di dalam atribut **for** selama nilai yang ditampilkan karena label adalah output dari field <outputText> yang terlampir didalamnya/
- <outputText> - hanya me-render text di dalam atribut nilainya menjadi yang sesuai dengan HTMLnya.
- <inputText> - me-render sebuah elemen input HTML berjenis text.
- <inputSecret> - me-render sebuah elemen input HTML berjenis password.
- <commandButton> - me-render sebuah elemen input HTML berjenis submit.

Dibawah adalah screenshot output hasil dari halaman JSP diatas.



Dan berikutnya adalah source HTML dari output.

```
<HTML>
    <TITLE>Login Page</TITLE>
    <BODY>

    Please login with your login name and password : <br/><br/>

   <form id=id="simpleForm" method="post" action="/JSFApplication/index.jsf"
enctype="application/x-www-form-urlencoded">

        <label for="simpleForm:loginName">
            Login Name:
        </label>
        <input id="simpleForm:loginName" type="text" name="simpleForm:loginName" /><br/>

        <label for="simpleForm:password">
             Password :
        </label>

        <input id="simpleForm:password" type="password" name="simpleForm:password"
value="" /><br/>

        <input type="submit" name="simpleForm:_id5" value="Login" />
        <input type="hidden" name="simpleForm" value="simpleForm" /></form>

    </BODY>
</HTML>
```

Kita dapat melihat dari listing HTML diatas bagaimana implementasi JSF render kompone-komponen Kita seperti dirumuskan di dalam tag-tag. Elemen form dikenalkan ke penggunaan method POST dari submission form, dengan action yang menunjuk pada halaman yang sama. Juga, mengenali bagaimana nilai id elemen HTML yang sebelum ditentukan dengan nama dari form. Hal ini memastikan bahwa nama elemen form bersifat unik di dalam aplikasi, dimana yang penting bagi operasi Faces.

9.1.3.2 MENGIKAT NILAI

Komponen UI yang menyusun view diatas memerlukan sebuah backing model untuk dapat menyimpan data yang ingin dimasukkan. Implementasi backing model ini telah didiskusikan sebelumnya. Satu-satunya pertanyaan adalah bagaimana caranya menghubungkan komponen Kita ke backing model.

Potongan JSP diatas akan dibuat ulang disini untuk pemanggilan ulang yang lebih mudah.

```
...
        <h:inputText id="loginName" value="#{loginPage.loginName}"/><br/>
        <h:outputLabel for="password">
            <h:outputText value="Password : "/>
        </h:outputLabel>
        <h:inputSecret id="password" value="#{loginPage.password}"/><br/>
...
```

Perhatikan text yang ditulis tebal. Notasi # yang bertindak sebagai nilai untuk atribut **value** yang mengikat properties di dalam LoginPageBean Kita untuk komponen UI Kita, dengan komponen menjadi batas property loginName dan komponen inputSecret menjadi batas untuk property password. **LoginPage** dalam hal ini mengacu kepada sebuah instans LoginPageBean yang akan menyimpan data.

Halaman JSF mampu untuk menghubungkan identifier loginPage kepada sebuah instans dari suatu LoginPageBean karena isi dari faces-config.xml. Masukan relevan diperkenalkan dibawah :

```
   <managed-bean>
      <description>
         This bean serves as the backing model for our login form
      </description>
      <managed-bean-name>loginPage</managed-bean-name>
      <managed-bean-class>jedi.sample.LoginPageBean</managed-bean-class>
      <managed-bean-scope>request</managed-bean-scope>
...
```

Karena LoginPageBean dideklarasikan untuk mengatur bean di dalam framework, sebuah instans LoginPageBean akan dibuat dan diletakkan di dalam scope konfigurasi (jika salah satunya tidak ada) ketika halaman JSF dievaluasi. Nilai yang diberikan oleh user kemudian diletakkan kedalam ikatan properties yang sesuai.

9.1.3.3 MENDAFTARKAN ACTION HANDLERS KE KOMPONEN VIEW

Di dalam listing HTML sebelumnya, Kita dapat melihat bahwa form Kita, ketika direnderer dengan tag custom JSF, poin-poin hanya untuk diri sendiri dengan atribut actionnya. Aplikasi web tradisional akan menempatkan URL atau mapping URL komponen yang akan menangani submission form di dalam atribut action. Benar-benar, Faces menangani aspek ini dengan cara yang berbeda.

JSF mengenalkan konsep pemrograman berbasis event ke dalam lingkungan web. Setiap komponen UI yang akan JSF sediakan diberi action user yang sesuai atau input, meng-generate events dapat diproses oleh action handlers.

Di dalam JSF di atas, sebagai ganti pemikiran bahwa menekan tombol submit akan mengakibatkan submission form, pikirkan bahwa menekan tombol akan mengakibatkan sebuah ActionEvent di-generate, dimana kemudian dapat diproses oleh sebuah handler.

Meninjau lagi ke dalam halaman JSF :

```
...
<h:commandButton action="#{loginPage.performLogin}" value="Login"/>
...
```

Kita temukan sebuah notasi # serupa dengan yang digunakan dalam mengikat sebuah property bean untuk sebuah komponen UI. Niat disini sama seperti : Kita mengikat sebuah method dengan nama performLogin yang ditemukan di dalam sebuah bean yang disesuaikan dengan nama loginPage kepada tombol komponen UI Kita. Sekarang, sebagai ganti penyimpanan nilai untuk komponen, method batas beraksi sebagai action handler untuk media tombol.

Ketika user meng-klik pada tombol ini, framework memanggil action handler yang ditunjuk. Lalu menggunakan hasil String dari action handler untuk menentukan halaman mana yang akan pergi ke yang berikutnya dengan mencarinya di dalam aturan navigasi yang dikonfigurasikan.

Untuk mendapatkan suatu pengertian mendalam yang lebih baik di komponen-komponen yang JSF sediakan, akan menjadi lebih baik ketika melihat pada komponen Swing

- Setiap komponen Swing menjelaskan sebuah widget interface, masing-masing dengan kemampuannya yang dapat diakses secara program menggunakan API. Contoh : sebuah komponen JTextField menampilkan sebuah field text yang dapat menerima input dari user. Nilai dapat diterima menggunakan getText().
- Masing-masing komponen juga berbagi beberapa properties dan method yang bersifat umum untuk semua komponen yang lain(Sebagai contoh, sebuah method field text setName()).
- Beberapa komponen Swing adalah Container, dimana mereka dapat menerima komponen-komponen lain untuk membentuk suatu gabungan yang kemudian dapat ditampilkan kepada user atau handler. Contoh : JPanel, JFrame keduanya dapat digunakan untuk diisikan oleh komponen-komponen lain.

Komponen Swing bersifat berbasis event dan interaksi user dengan komponen Swing mengakibatkan generate sebuah event yang dilaksanakan oleh listener-listener yang diregister.