www.tobuku.com

MODEL MVC DENGAN STRUTS

Desember 2009
Oleh : Feri Djuandi

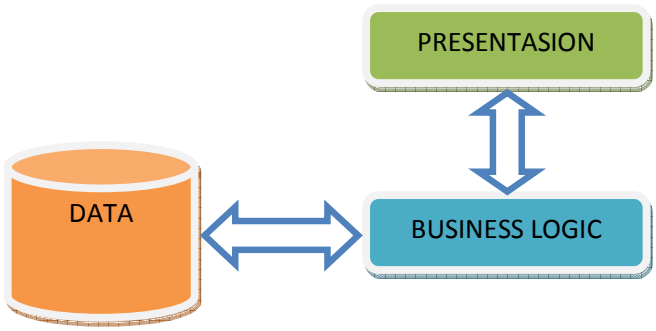
Tingkat: ☐ Pemula ☑ Menengah ☐ Mahir

Apakah Struts itu?

Secara singkat **Struts** dapat didefinisikan sebagai sebuah *web application framework* untuk platform Java Enterprise Edition (Java EE). Strust yang dibahas pada dokumen ini adalah **versi 2.0**, yang merupakan pengembangan selanjutnya dari versi sebelumnya dengan beberapa perubahan yang signifikan.

Framework adalah sebuah software struktural, yang wujudnya berupa *object library* – bisa berupa hasil kompilasi (*binary code*) atau kode program (*source code*). Pada prinsipnya tujuan yang ingin dicapai oleh kehadiran sebuah framework adalah "*don't reinvent the wheel*". Artinya di dalam sebuah framework sudah tersedia bermacam-macam komponen yang jumlahnya mungkin mencapai ratusan atau bahkan ribuan untuk berbagai fungsi dan keperluan sehingga bisa langsung dipakai oleh seorang programmer di dalam mengembangkan sebuah aplikasi. Tanpa menggunakan sebuah framework, programmer tersebut tentu harus menghabiskan banyak waktu dan tenaga untuk membuat komponen-komponen itu termasuk melakukan tes dan menerima resiko bahwa komponen yang dibuatnya bisa saja mengandung cacat (*bug*). Komponen-komponen yang ada di dalam sebuah framework umumnya teruji dengan baik dan handal karena sudah melewati rentang waktu cukup lama dalam pemakaian, uji coba serta perbaikan di dalam berbagai penerapan, dan tentunya mulai dari tahap desain hingga pengembangan melibatkan banyak masukan dari para kontributor yang berpengalaman dan berpemikiran matang. Jadi apa yang ada di dalam sebuah framework adalah hasil kolektif dari para pakar yang ahli di bidangnya. Tanpa mengurangi rasa hormat dan penghargaan atas kerja keras programmer tadi, hasil kerja puluhan hingga ratusan pakar dan dukungan jutaan orang dalam komunitas sungguh tidak bisa diperbandingkan dengan hasil kerja seorang programmer. Jadi untuk tujuan efisiensi dan produktifitas, seorang programmer perlu mempertimbangkan dengan serius penggunaan framework di dalam pekerjaannya.

Sebelum membahas Struts lebih lanjut ada baiknya Anda memperhatikan gambar sederhana di bawah ini yang mengilustrasikan konsep aplikasi *3-tier* untuk mendapatkan pengertian di mana sesungguhnya posisi Struts dalam arsitektur ini.

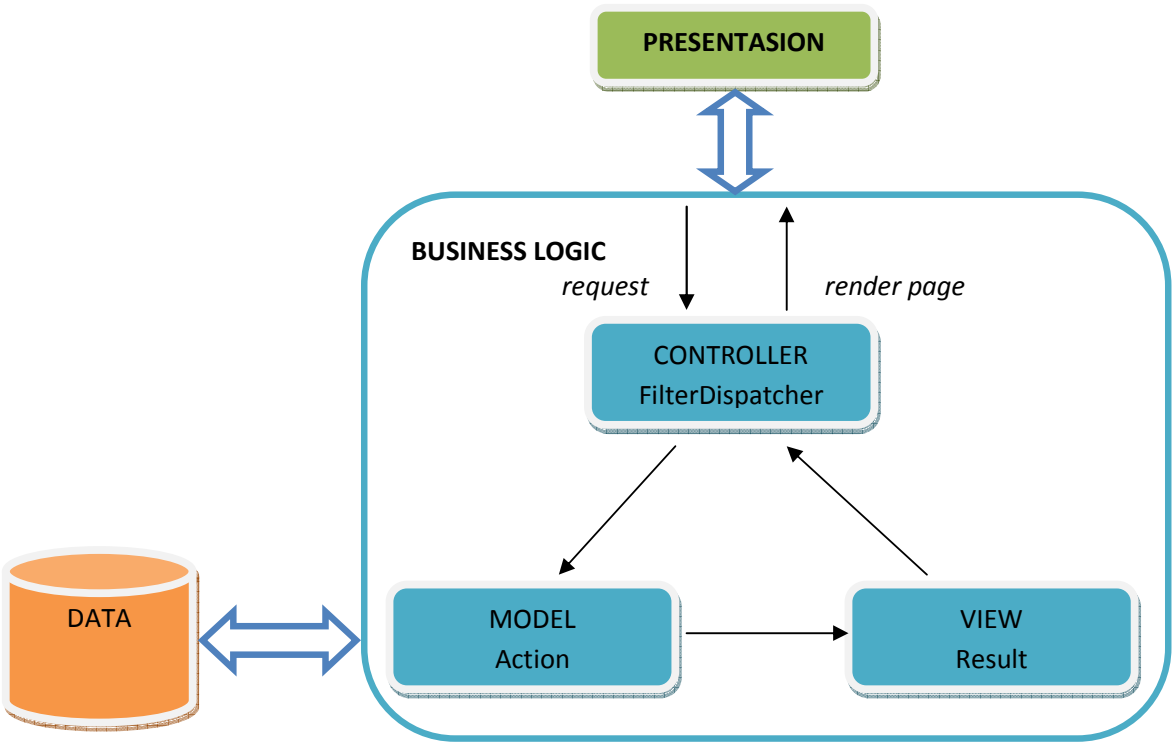


Gambar 1.

Tanpa perlu dijelaskan panjang lebar, Anda tentunya paham bahwa *presentation layer* berhubungan dengan antara muka yang menjadi fasad dari sebuah aplikasi dimana para pengguna bisa berinteraksi dengan sistem di dalamnya. Seorang pengguna tidak akan peduli bagaimana program bekerja di belakang layar yang baginya adalah sebuah kotak hitam – yang dia peduli adalah tampilan-tampilan angka dan tulisan kasat mata yang muncul di layer komputernya.

Data layer berkaitan dengan media penyimpanan data dari aplikasi atau lebih dikenal sebagai database. Bentuknya bisa berupa database SQL Server, DB-2, Oracle dan sebagainya. Sementara itu, *business logic* adalah inti dari sebuah aplikasi karena ia yang mengendalikan seluruh jalannya sistem. Semua aliran program, validasi, pemrosesan data, sekuriti dan sebagainya terdapat di dalam lapisan ini. Struts ada di dalam lapisan business logic (kadang-kadang disebut juga *domain logic* atau *application logic*) – tepatnya Struts menjadi pondasi bagi kode program di dalam business logic. Inilah yang perlu dipahami mengenai posisi Struts. Struts tidak ada kaitannya dengan presentation layer dan data layer – namun ketidak-bergantungannya membuat Struts menjadi fleksibel untuk dipasangkan dengan bermacam-macam teknologi misalnya Struts dapat dikolaborasikan dengan iBATIS, Hybernate atau Spring.

Pada gambar di atas tampak bahwa business logic secara asitektural dibagi menjadi tiga bagian, yaitu *controller*, *model* dan *view*. Struts adalah web application framework yang mengimplementasikan pola desain **Model–View–Controller** (**MVC**). Pola ini dengan tegas mengisolasi domain logic dari presentation layer yang akan membuat komponen-komponen aplikasi menjadi independen baik dalam hal pengembangan, testing maupun pemeliharaan. MVC adalah sebuah pendekatan yang umum dan tidak terkait dengan teknologi atau produk tertentu. Konsep MVC sudah diadopsi secara luas dalam berbagai pengembangan software. Pola MVC ini harus dipahami dengan baik karena kunci pemahaman Struts adalah MVC. Jika Anda mengerti konsep MVC maka Anda sudah separuh jalan menguasai Struts – semudah itu. Sesungguhnya MVC adalah konsep yang cukup sederhana dan tidak sulit dipelajari. Mari kita kupas satu per satu dimulai dari controller.



Gambar 2.

CONTROLLER

Tugas controller adalah memetakan *request* (permintaan) dengan *action* (respon). Pekerjaannya mirip seorang pelayan di sebuah restoran. Pada jam-jam sibuk saat restoran penuh dengan sejumlah tamu, seorang pelayan akan menerima banyak pesanan makanan. Kemudian ia akan meneruskan pesanan itu ke bagian dapur, sementara itu dari bagian dapur ia akan menerima makanan-makanan yang baru selesai dimasak dan meneruskannya kepada tamu yang memesannya. Jadi, sebuah controller adalah komponen yang menerima request dari presentation layer dan meresponnya melalui sebuah action. Pada Struts 2, controller diperankan oleh **FilterDispatcher**. FilterDispatcher adalah sebuah object penting yang berwujud *servlet filter* yang menginspeksi setiap request yang datang untuk menentukan action mana yang harus dilakukan.

MODEL

Controller memang memiliki pekerjaan yang sangat sibuk untuk melayani banyak request, namun yang dilakukannya adalah semata-mata tugas administratif, bukan tugas inti. Tugas inti itu sendiri dilakukan oleh model. Model adalah komponen utama di dalam domain logic karena sesungguhnya model adalah aplikasi itu sendiri. Di dalam pola MVC, model diimplementasikan dalam wujud Javabean (atau lebih sering cukup disebut bean).

VIEW

Yang terakhir yaitu view adalah presentasi atau hasil dari action – sesuatu yang ingin disajikan kepada presentation layer sebagai respon dari request yang diajukannya. Umumnya view berbentuk halaman JSP, walaupun bentuk hasil lainnya pun dimungkinkan, misalnya file HTML, PDF, ZIP, gambar dan sebagainya.

Sebelum Anda menjadi bingung, sebaiknya teori mengenai MVC diakhiri dulu sampai di sini. Akan lebih baik bila penjelasan ini langsung disertai dengan praktek sehingga Anda bisa mendapatkan gambaran yang jauh lebih baik mengenai topik ini.

Persiapan Sebelum Memulai

Sebelum melanjutkan pembahasan Struts, silakan mendapatkan library Struts yang akan dibutuhkan selama penjelasan di dalam artikel ini. Library Struts tersedia secara gratis dengan cara men-download-nya melalui situs web:

http://struts.apache.org/

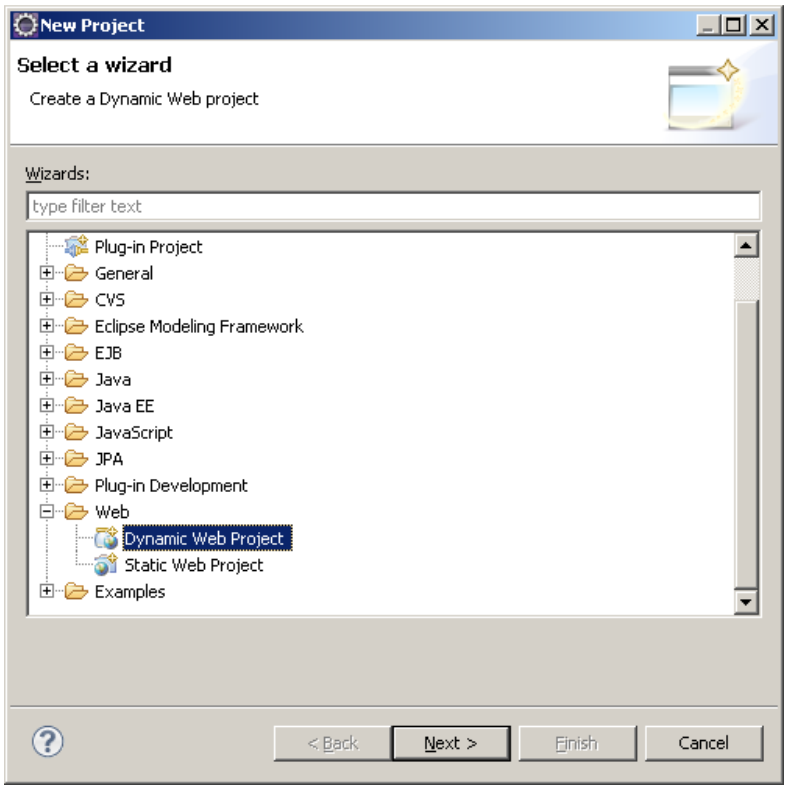
Pada artikel ini versi Struts yang digunakan adalah 2.1.8.1. Anda memiliki kebebasan untuk menggunakan versi yang lebih baru. Kecuali ada hal-hal yang disebutkan secara khusus, seharusnya contoh-contoh program di dalam artikel ini dapat berjalan dengan baik pada Struts versi yang lebih baru.

Paket library Struts umumnya berbentuk file ZIP yang di dalamnya terdapat file-file library, dokumentasi dan contoh program. Silakan ekstrak file ZIP tersebut. Library Struts dapat ditemukan di dalam folder **lib**, sementara folder-folder yang lain memuat contoh program dan dokumentasi.

Mengasosiasikan Project dengan Struts Framework

Setelah langkah-langkah persiapan di atas selesai dilakukan, maka Struts bisa mulai digunakan.

1. Jalankan **Eclipse** dan pilih menu **File → New → Project**. Pada wizard, pilihlah "**Dynamic Web Project**".



2. Beri nama project tersebut sebagai **mvc-exercise01**. Ikuti langkah-langkah pada wizard hingga selesai.

3. Dengan menggunakan Windows Explorer, copy file-file berikut ini dari folder **lib** hasil ekstraks library Struts ke dalam folder **..\ WebContent\WEB-INF\lib** dari folder project yang bersangkutan.

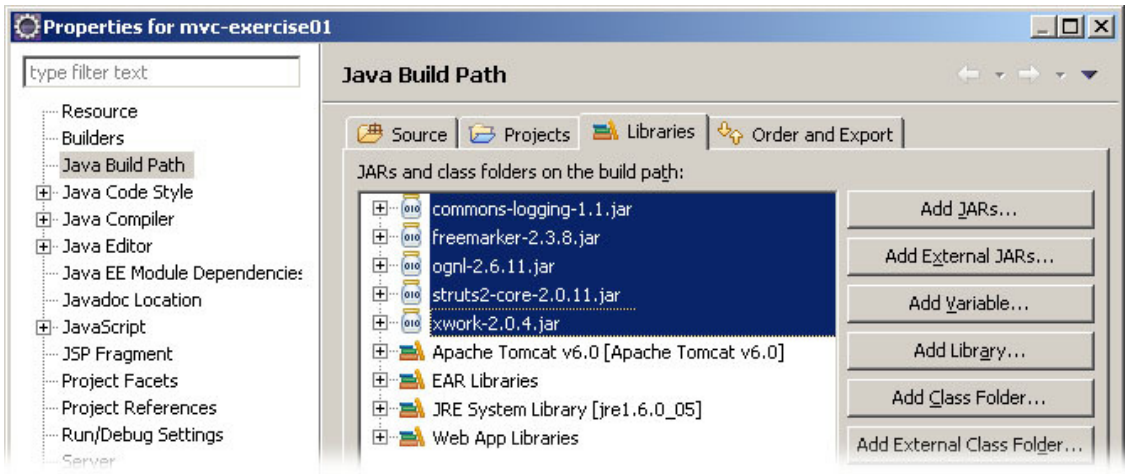
   - commons-io-1.3.2.jar
   - commons-logging-1.0.4.jar
   - freemarker-2.3.15.jar
   - ognl-2.7.3.jar
   - struts2-core-2.1.8.1.jar
   - xwork-core-2.1.6.jar

   Nama file-file di atas bisa berbeda sesuai dengan versi Struts yang digunakan.

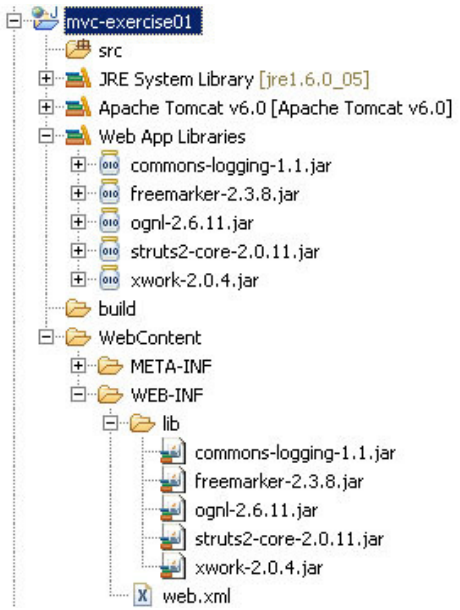
File-file di atas adalah beberapa bagian dari Struts framework yang dibutuhkan oleh contoh program ini. Tidak semua file library Struts perlu di-copy ke dalam sebuah project karena hanya file-file library tertentu saja yang relevan dengan peruntukannya. Untuk mendapatkan informasi mengenai file-file library dan kegunaannya silakan membaca dokumentasi selengkapnya yang ada di dalam paket Struts.

4. Kembali kepada Eclipse, klik-kanan pada folder mvc-exercise01 kemudian pilih menu **Build Path → Configure Build Path**.

   Klik tombol "**Add External JARs**" dan pilih file-file yang telah di-copy ke dalam folder lib tadi.



5. Tekan F5 untuk me-refresh tampilan pada window Package Explorer. Jika file-file library telah ditambahkan, maka tampilan pada project akan tampak seperti di bawah ini.



Mengedit web.xml

Setelah mengasosiasikan library Struts dengan project, maka langkah berikutnya adalah mengatur konfigurasi Struts yang ada di dalam file web.xml. Buka file web.xml yang ada di dalam folder WEB-INF. File web.xml disebut sebagai **deployment descriptor**. File web.xml adalah file esensial yang harus ada di dalam sebuah aplikasi web dan di dalamnya terdapat definisi servlet dan servlet filter yang berkaitan dengan aplikasi web yang bersangkutan. Contoh sebuah web.xml diperlihatkan di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
  <display-name>mvc-exercise01</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

File ini harus diedit dengan menambahkan definisi servlet filter dari Struts berikut ini.

```
    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
   </filter>

    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Hasil modifikasi file web.xml akan tampak seperti di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
      id="WebApp_ID" version="2.5">
      <display-name>mvc-exercise01</display-name>

      <filter>
            <filter-name>struts2</filter-name>
            <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
      </filter>

      <filter-mapping>
            <filter-name>struts2</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>

      <welcome-file-list>
```

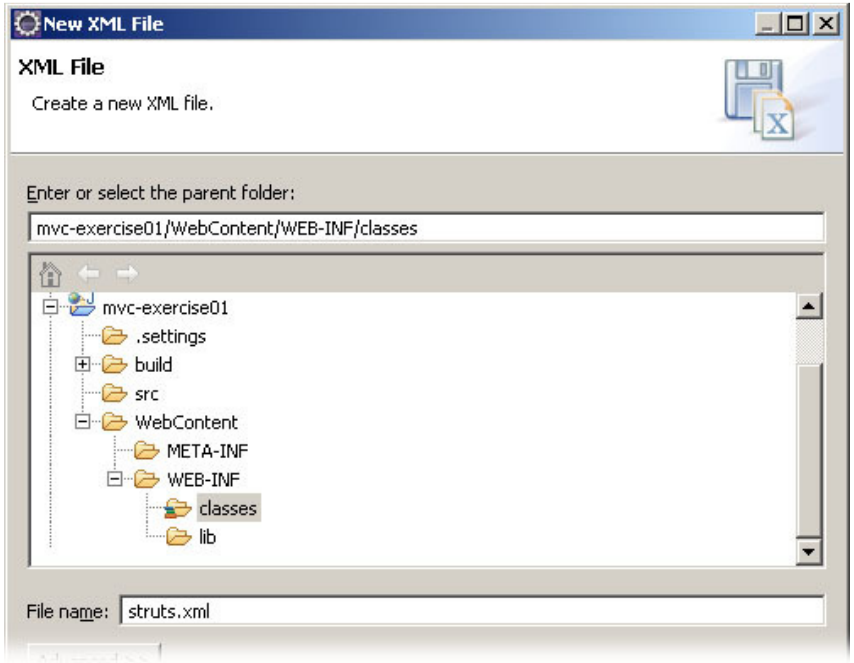
```
            <welcome-file>index.html</welcome-file>
            <welcome-file>index.htm</welcome-file>
            <welcome-file>index.jsp</welcome-file>
            <welcome-file>default.html</welcome-file>
            <welcome-file>default.htm</welcome-file>
            <welcome-file>default.jsp</welcome-file>
      </welcome-file-list>
</web-app>
```

File web.xml di atas adalah format yang standar untuk penggunaan Struts sehingga file ini bisa disimpan untuk digunakan pada aplikasi-aplikasi web yang lain tanpa perlu diubah, kecuali bagian tag <display-name>.

Menambahkan struts.xml

Merujuk pada gambar nnn yang memperlihatkan pola desain MVC, maka saat ini langkah selanjutnya yang akan dilakukan adalah mempersiapkan controller. *Pada Struts, controller diimplementasikan dalam bentuk file yang bernama* **strust.xml**. File ini akan berisi informasi-informasi action serta pasangan-pasangan model dan view-nya.

1. Buat sebuah sub-folder di dalam folder WEB-INF melalui menu **New → Folder** dan beri nama sebagai **classes**.
2. Buat sebuah file XML di dalam sub-folder classes melalui menu **New → Other**, kemudian pilih **XML**. Beri nama file tersebut sebagai **struts.xml**.



Edit file XML tersebut dengan menambahkan beberapa baris kode seperti berikut ini.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
      <constant name="struts.devMode" value="true" />

      <package name="default" namespace="/" extends="struts-default">
            <action name="HelloStruts" class="">
                  <result>/Hello.jsp</result>
            </action>
      </package>
</struts>
```

File strust.xml di atas adalah bentuk yang sangat sederhana. Isi dari file tersebut dapat dijelaskan sebagai berikut:

- `<constant name="struts.devMode" value="true" />`

  Struts memiliki sebuah setting yang bernama development mode (devMode) yang bisa di-set true atau false (fault-nya adalah false). Setting ini dianjurkan pada saat development karena akan memudahkan programmer melakukan troubleshot jika terjadi kesalahan. Baris ini boleh dihilangkan jika aplikasi sudah di-test dengan baik dan siap dipromosikan ke production.

- ```
  <package name="default" namespace="/" extends="struts-default">
        ...

  </package>
  ```

  Package adalah cara untuk mengelompokkan action, result, interceptor (interceptor akan dibahas kemudian) dan sebagainya secara logikal di dalam sebuah unit konfigurasi. Bentuk penamaan package di atas adalah format yang standar dan umum digunakan.

  Berbeda dengan nama sebuah package yang tujuannya adalah sebagai referensi dari sebuah unit konfigurasi, nama sebuah namespace perlu mendapat perhatian karena menentukan pemanggilan action. Penggunaan nama namespace pada file struts.xml serupa dengan
  ```

penggunaan namespace pada kode program Java yang tujuannya menghindari konflik dari nama-nama yang sama. Jika pada program Java yang ingin dihindari adalah konflik penamaan class, maka pada struts.xlm yang ingin dihindari adalah konflik dari nama action yang sama. Nama namespace pada sebuah package adalah opsional, maksudnya nama namespace tidak perlu disebutkan misalnya seperti berikut ini:

```
<package name=" mypackage" extends="struts-default">

...
```

Jika demikian halnya, maka namespace yang digunakan adalah kosong atau disebut sebagai **default namespace**. Sementara itu, nama namespace yang berbentuk namespace="/" disebut **root namespace** yang artinya ia ada di posisi teratas pada hirarki namespace. Jika sebuah namespace ingin diberi nama, maka format bisa berbentuk seperti di bawah ini:

```
<package name="mypackage" namespace="/mynamespace" extends="struts-default">

...
```

- ```
  <action name="HelloStruts" class="">
        <result>/Hello.jsp</result>

  </action>
  ```

  Nama action di dalam blok package merujuk pada respon dari sebuah request yang ingin dieksekusi. Nama sebuah action harus unik di dalam sebuah namespace karena ia menjadi referensi yang spesifik untuk setiap request. Nama class merujuk pada nama sebuah bean yang akan dijalankan menyusul pemanggilan action yang bersangkutan. *Dalam hal ini sebuah bean merepresentasikan sebuah model di dalam desain MVC*. Pada contoh ini nama class sengaja dikosongkan karena kita belum akan membuat sebuah business logic. Terakhir, *result di dalam blok action mewakili sebuah view,* yaitu sesuatu yang ingin dikembalikan kepada presentation layer.

  Sebagai kesimpulan, struts.xml adalah salah satu bagian terpenting di dalam Struts framework karena file ini berisi konfigurasi untuk controller, model dan view.

Untuk memberikan gambaran lebih jelas mengenai namespace dan action, berikut ini adalah sebuah contoh lain. Pada kode di bawah ini tampak tiga buah package dimana masing-masing namespace-nya adalah default namespace, root namespace dan sebuah namespace bernama barspace. Perhatikan sebuah action bernama bar yang ada di dalam default namespace dan barspace. Sekalipun namanya sama, namun kedua action ini memiliki namespace yang berbeda sehingga mereka tidak akan konflik satu sama lain.
  ```

```
<package name="default">
    <action name="foo" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">greeting.jsp</result>
    </action>

    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar1.jsp</result>
    </action>
</package>

<package name="mypackage1" namespace="/">
    <action name="moo" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">moo.jsp</result>
    </action>
</package>

<package name="mypackage2" namespace="/barspace">
    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar2.jsp</result>
    </action>
</package>
```

Jika ada sebuah request terhadap action /barspace/bar maka namespace barspace akan dirujuk terlebih dahulu untuk mendapatkan action bar. Jika ditemukan, maka action bar dieksekusi, tetapi *jika action tidak ditemukan di dalam sebuah namespace maka action tersebut akan dicari di dalam default namespace*. Pada contoh ini, action bar ditemukan di dalam namespace barspace sehingga jika hasil yang dikembalikan adalah "success" maka request akan diteruskan kepada bar2.jsp.

Jika ada sebuah request terhadap /barspace/foo.action, maka sekali lagi namespace barspace akan dirujuk untuk mendapatkan action foo. Karena action foo tidak ditemukan di dalam namespace barspace, maka pencarian akan dilanjutkan kepada default namespace.

Untuk memberikan gambaran lebih jelas mengenai namespace dan action, berikut ini adalah sebuah contoh lain. Pada kode di bawah ini tampak tiga buah package dimana masing-masing namespace-nya adalah default namespace, root namespace dan sebuah namespace bernama barspace. Perhatikan sebuah action bernama bar yang ada di dalam default namespace dan barspace. Sekalipun namanya sama, namun kedua action ini memiliki namespace yang berbeda sehingga mereka tidak akan konflik satu sama lain.

```
<package name="default">
    <action name="foo" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">greeting.jsp</result>
    </action>

    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar1.jsp</result>
    </action>
</package>

<package name="mypackage1" namespace="/">
    <action name="moo" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">moo.jsp</result>
    </action>
</package>

<package name="mypackage2" namespace="/barspace">
    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar2.jsp</result>
    </action>
</package>
```

Jika ada sebuah request terhadap action /barspace/bar maka namespace barspace akan dirujuk terlebih dahulu untuk mendapatkan action bar. Jika ditemukan, maka action bar dieksekusi, tetapi *jika action tidak ditemukan di dalam sebuah namespace maka action tersebut akan dicari di dalam default namespace*. Pada contoh ini, action bar ditemukan di dalam namespace barspace sehingga jika hasil yang dikembalikan adalah "success" maka request akan diteruskan kepada bar2.jsp.

Jika ada sebuah request terhadap /barspace/foo.action, maka sekali lagi namespace barspace akan dirujuk untuk mendapatkan action foo. Karena action foo tidak ditemukan di dalam namespace barspace, maka pencarian akan dilanjutkan kepada default namespace.

Menambahkan Hello.jsp

Mengacu pada kode program nnnn, tampak bahwa action HelloStruts akan menghasilkan sebuah view bernama Hello.jsp.

Buat sebuah file JSP di dalam folder WebContent melalui menu **New → Other**, kemudian pilih **Web → JSP**. Beri nama file tersebut sebagai **Hello.jsp**.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Hello in Action</title>
</head>
<body>
Hello Struts!!
</body>
</html>
```

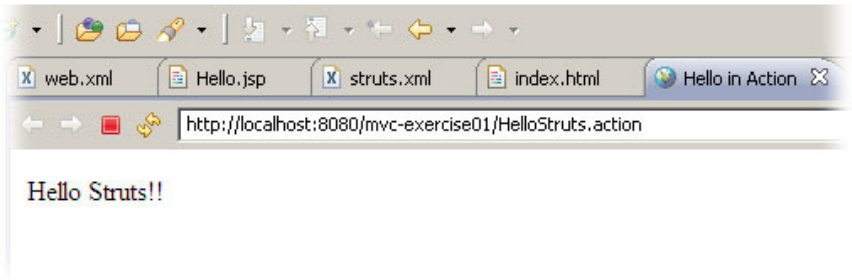
Menambahkan index.html

Sebagai langkah terakhir pada contoh ini, kita memerlukan sebuah rutin yang akan memicu pemanggilan sebuah action. Pada kali ini kita akan melakukannya melalui file index.html.

Buat sebuah file HTML di dalam folder **WebContent** melalui menu **New → Other**, kemudian pilih **Web → HTML Page**. Beri nama file tersebut sebagai **index.html**.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=HelloStruts.action">
<title>Hello Struts</title>
</head>
<body>
<h3>One moment please.</h3>
</body>
</html>
```

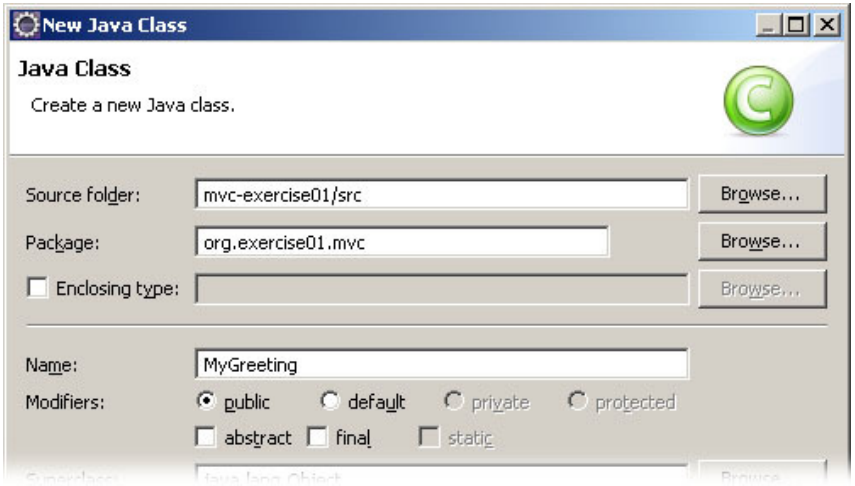
Silakan simpan semua perubahan di atas, lalu jalankan program tersebut. Tampilan dari program tampak seperti gambar di bawah ini.



Lebih Lanjut dengan Model

Sejauh ini contoh program yang dibuat belum melibatkan sebuah model. Pada contoh berikutnya akan diperlihatkan sebuah bean yang dibuat untuk mewakili sebuah domain logic.

1. Pada Eclipse buatlah sebuah class di dalam foder **src**. Tentukan nama package-nya sebagai **org.exercise01.mvc**, sedangkan nama class-nya adalah **MyGreeting**.



2. Ketikkan kode program di bawah ini ke dalam class tersebut.

```
package org.exercise01.mvc;

public class MyGreeting {
	private static final String GREETING = "Hello ";
	private String name;
	private String customGreeting;

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getCustomGreeting() {
```

```
		return customGreeting;
	}
	public void setCustomGreeting(String customGreeting) {
		this.customGreeting = customGreeting;
	}

	public String execute()  {
		setCustomGreeting( GREETING + getName() );
		return "SUCCESS";
	}
}
```

3. Edit file **struts.xml** dan buat sebuah package baru bernama mypackage seperti di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
	<constant name="struts.devMode" value="true" />

	<package name="default" namespace="/" extends="struts-default">
		<action name="HelloStruts" class="">
			<result>/Hello.jsp</result>
		</action>
	</package>

	<package name="mypackage" namespace="/myspace" extends="struts-default">
		<action name="GreetingStruts" class="org.exercise01.mvc.MyGreeting">
			<result name="SUCCESS" type="dispatcher">/Greeting.jsp</result>
		</action>
	</package>

</struts>
```

4. Buat file **Greeting.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Greeting Struts</title>
</head>
<body>
<h3>Custom Greeting Page</h3>
<h4><s:property value="customGreeting" /></h4>

</body>
</html>
```

5. Edit file Hello.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Hello in Action</title>
</head>
<body>
Hello Struts!!
<hr>
Enter your name so that we can customize a greeting just for you!
<s:form action="myspace/GreetingStruts.action">
      <s:textfield name="name" label="Your name" />
      <s:submit />
```

```
</s:form>
<hr>

</body>
</html>
```

Silakan simpan semua perubahan di atas, lalu jalankan program tersebut. Tampilan dari program tampak seperti gambar di bawah ini.