Struts y JavaServer Faces, cara a cara.

Francisco J. Díaz, Claudia A. Queiruga, Laura A. Fava
LINTI – Facultad de Informática
La Plata, Buenos Aires, Argentina.
{jdiaz, claudiaq, lfava}@info.unlp.edu.ar

1. CONTEXTO

Nuestra investigación forma parte de una del área "Desarrollo de Aplicaciones", del proyecto "Redes, Seguridad y Desarrollo de Aplicaciones para e-educación, e-salud, e-gobierno y e-inclusión" del Laboratorio de Investigación en Nuevas Tecnologías Informáticas, LINTI.

2. RESUMEN

J2EE [Ref.1] es una plataforma para desarrollo de aplicaciones empresariales, escalable, robusta, multiplataforma, madura y muy bien documentada. A pesar de esto, desde el punto de vista del programador escribir aplicaciones usando la API[1] de Servlets, JSP y EJB es una tarea tediosa, poco amigable y de baja productividad. Por otro lado, la aplicación resultante, en términos generales es poco estructurada, basada en componentes de baja reusabilidad y difícil de mantener.
Los *frameworks* J2EE facilitan el desarrollo de aplicaciones empresariales, reducen el tiempo involucrado en el proceso de desarrollo y mejoran notablemente la calidad del software resultante. Los programadores pueden dedicarse a resolver los problemas específicos de la lógica del negocio, dejando de lado los detalles de programación de bajo nivel.
El objetivo de este artículo es comparar 2 *frameworks* J2EE de código fuente abierto: Struts [Ref.2, 3] y JavaServer Faces (JSF) [Ref.4] y establecer pautas que faciliten la elección a la hora de decidir cuál es el *framework* más adecuado para la implementación de un proyecto determinado.

Palabras claves: frameworks J2EE, Struts, JavaServer Faces, MVC paraWeb, componentes de IU para web (UI Web Component).

3. FORMACIÓN DE RECURSOS HUMANOS

Los integrantes de esta línea de investigación dirigen a un docente con semi-dedicación del LINTI, cuyo tema de investigación es **"Desarrollo de nuevas componentes sobre el framework JFS"**, a un becario del LINTI que está desarrollando su Tesina de Grado de Licenciatura en Informática en temas vinculados a desarrollo de aplicaciones.
Además, dos de los integrantes de esta área de investigación, son docentes del curso de posgrado **"Desarrollo de aplicaciones usando J2EE"**, en la Facultad de Informática de la UNLP, y de la cátedra **"Java y Aplicaciones Avanzadas en Internet"**, de quinto año de la carrera Lic. en Informática, en donde los alumnos realizan un trabajo final, basado en el desarrollo de componentes JSF nuevas y *customizadas* .

4. INTRODUCCIÓN

4.1. ¿QUÉ ES UN FRAMEWORK WEB J2EE?

Un *framework*, es una estructura software compuesta de componentes *personalizables* e intercambiables que permiten desarrollar una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que podemos agregarle algunas piezas para construir una aplicación concreta.

[1] API Application Programming Interface, en castellano interface de programación de aplicaciones.

Los objetivos principales de los *frameworks* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.
En particular, un *framework web* J2EE, es un conjunto de componentes de software, por ejemplo clases JAVA, descriptores y archivos de configuración en XML, basados en la plataforma J2EE y, que constituyen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web que se ejecutarán en servidores J2EE.
Los dos *frameworks web* que compararemos en este artículo, responden al patrón de diseño MVC (Modelo Vista Controlador) para web, también conocido como MVC Modelo 2. Básicamente este patrón es una guía para el diseño de arquitecturas de aplicaciones que ofrecen una fuerte interactividad con usuarios. Organiza la aplicación en tres subsistemas separados: el **Modelo** que representa los datos de la aplicación y sus reglas de negocio, la **Vista** formada por un conjunto de pantallas que representa la interfaz de usuario de la aplicación (en web básicamente son formularios HTML de entrada y salida), y el **Controlador** que es el encargado de procesar las peticiones de los usuarios y controlar el flujo de ejecución del sistema. La Figura 1, ilustra las partes que constituyen una aplicación que responde al patrón MVC.

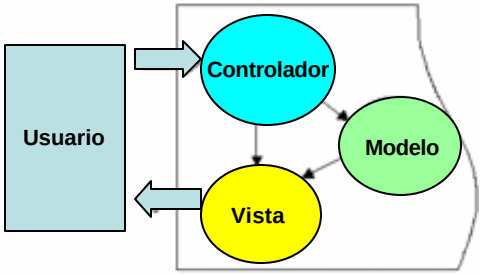

Figura 1. Arquitectura MVC

Los **frameworks web** que se comparan en este artículo, responden al patrón MVC para web y pertenecen a dos familias diferentes: frameworks orientados a la **interfaz de usuario**, como es **Java Server Faces** , y frameworks orientados al **control de eventos de petición**, como es **Struts**.

4.2. FUNDAMENTOS DEL FRAMEWORK STRUTS

Struts es un **framework web J2EE,** de la familia de software libre, implementa el patrón de diseño MVC2 o Modelo 2 y básicamente está construido sobre las tecnologías de Servlets y JSP [Ref 5]. Struts combina Servlets, JSP's, *custom tags* propios y recursos de la aplicación en un único *framework*. Fue creado por Craig McClanahan y forma parte del proyecto Apache Struts, del Apache Software Foundation [Ref.2].
Struts oculta al programador los detalles del protocolo HTTP, JSP, Servlets, etc. Un programador Struts puede desconocer estos nombres, sin embargo tener conocimiento de las tecnologías de base de Struts hace que se puedan hacer soluciones creativas.
El corazón de Struts es el Servlet Controlador (objeto **ActionServlet**), el cual intercepta todos los requerimientos HTTP entrantes, provenientes de los clientes y los delega a un manejador apropiado (objeto **Action**). Para determinar el flujo de la aplicación, es decir, a que Action pasar el requerimiento, hace uso del archivo **struts-config.xml**. El Servlet Controlador, luego recibe las respuestas de los Action y las redirecciona a la vista apropiada (**JSPs**). Para ello, nuevamente consulta un conjunto de mapeos definidos en el archivo de configuración xml.
Por último, existen múltiples objetos ActionForms (subclase de **ActionForm**). Estos objetos son JavaBeans [Ref.6] usados para mantener los datos ingresados por el usuario en las páginas JSP. Un

punto clave del *framework* Struts es que automáticamente llena los objetos **ActionForm** con datos de la petición del usuario.
Si bien Struts está basado en el patrón de diseño MVC2, solo provee componentes para las capas **Vista** y **Controlador**. Las componentes **Actions** y **ActionForms** son neutrales de la **Vista**, con lo cual, Struts puede ser usado con Velocity Templates [Ref.7], XSL [Ref.8] u otras tecnologías de presentación. Struts no provee componentes para el modelo, el cual generalmente se implementa con JavaBeans y EJB, pero esto no es mandatario, es de **modelo neutral**.
La Figura 2, ilustra las componentes claves del *framework* que participan en una aplicación Struts.

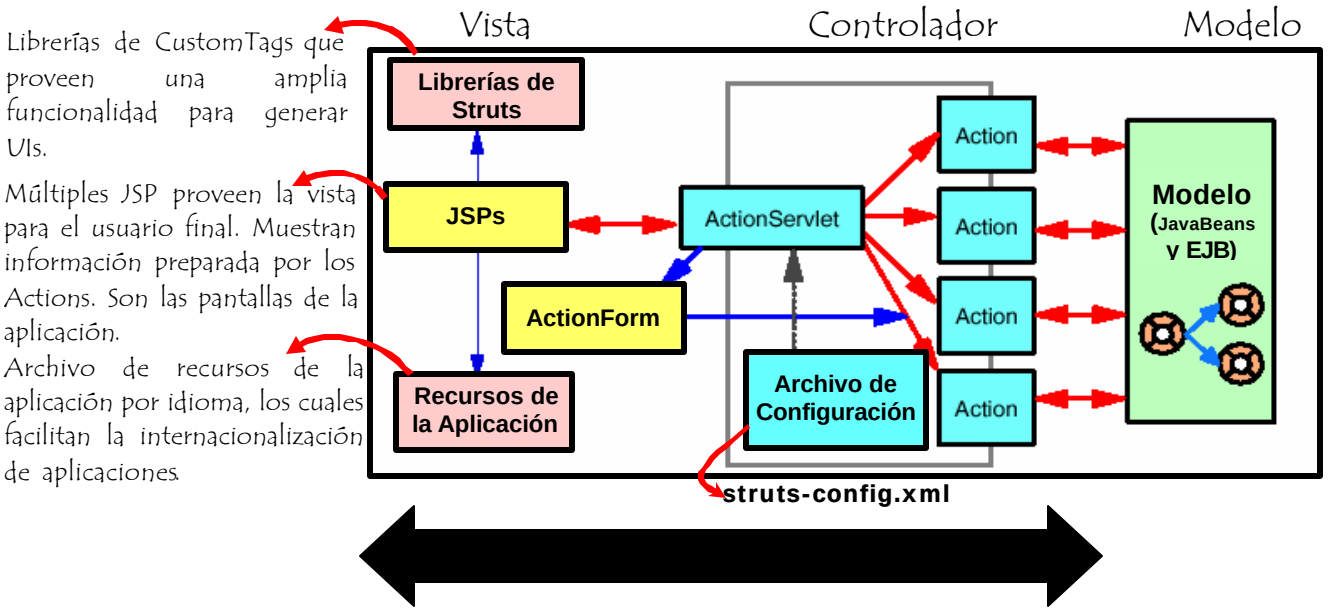

Figura 2. Arquitectura de una aplicación Struts

Struts está soportado en la mayoría de los IDE´s JAVA (Integrated Development Environmet) usados en la actualidad: Eclipse (http://www.eclipse.org/) mediante plugins específicos, JBuilder (de Inprise), WSAD (de IBM), JDeveloper (Oracle), etc.

4.3. FUNDAMENTOS DEL FRAMEWORK JAVASERVER FACES

A diferencia de Struts y de cualquier otro framework web J2EE, **JAVASERVER FACES (JSF)** es el único que tiene una especificación creada por el Java Community Process (JCP) [Ref.9], esto lo transforma en un estándar y como principal consecuencia todas las implementaciones, tanto las de código fuente abierto como las comerciales deben respetarla. Por otro lado, JSF 1.2 [Ref.10], la versión actual del framework, forma parte de la especificación J2EE 5.0 con lo cual todas las implementaciones de servidores J2EE 5.0 lo deben soportar.
JSF es un framework web J2EE de la familia de código fuente abierto, server-side, basado en componentes de interfaz de usuario con estado, que facilita y agiliza el desarrollo de aplicaciones web. Los programadores piensan en términos de **componentes de IU (interfaz de usuario)**, **eventos**, y **sus interacciones.** JSF oculta al programador los detalles del requerimiento HTTP, la respuesta HTTP y de *markups* HTML, logrando un estilo de programación similar a la programación de aplicaciones de escritorio, del estilo JAVA Swing, Delphi o Visual Basic [capítulo1, Ref.11].

JSF es una tecnología que permite construir aplicaciones web, que soportan diferentes dispositivos como clientes, por ej. teléfonos celulares, agendas, etc., no es una tecnología que soporta solamente clientes web browser, como lo es Struts. **Las componentes de IU son de *display* agnóstico**.
El framework JSF está construido solamente sobre la API de Servlets [Ref.5] y la distribución estándar provee un conjunto de librerías de *custom tags* JSP que permiten incluir las componentes de IU en páginas dinámicas. Sin embargo, es posible utilizar una tecnología de presentación diferente al de cliente web.
La Figura 3, muestra las componentes principales del framework JSF que intervienen en la construcción de una aplicación web J2EE, y también su flexibilidad para aceptar peticiones provenientes de diferentes clientes.

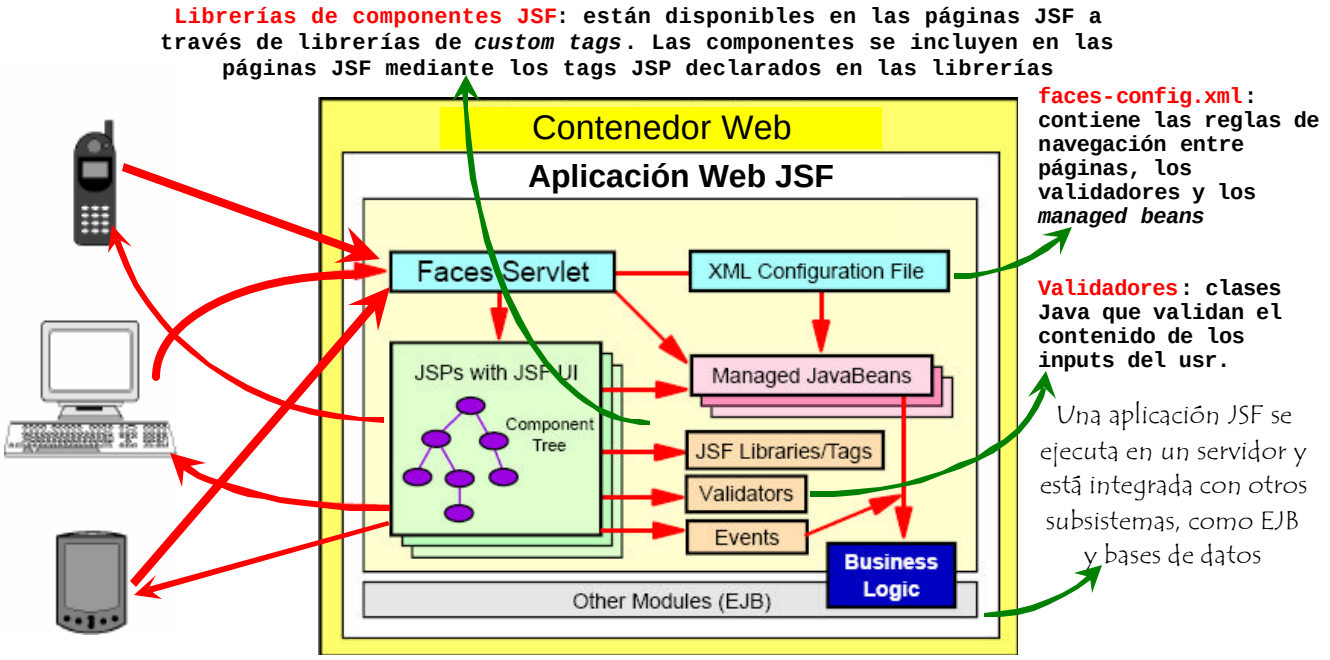

Figura 3. Arquitectura de una aplicación JSF

En forma análoga a Struts, JSF implementa el patrón Front-Controller, que centraliza el manejo de peticiones provenientes de los clientes. En el caso de JSF, este controlador central, es un objeto servlet, llamado **FacesServlet**.
Las aplicaciones JSF se construyen a partir de **componentes de IU**. La distribución estándar de JSF provee un conjunto de componentes básicas como botones, *checklist*, campos de entrada, etc., las que a su vez pueden ser *customizadas*. Las componentes de IU son **orientadas a eventos**, que se generan del lado del cliente y se procesan del lado del servidor, usando el modelo de eventos de la especificación de JavaBeans [Ref.6]. A su vez, las componentes proveen facilidades para **validación de entradas de datos** y **conversión** de objetos.

5. STRUTS VERSUS JAVASERVER FACES

5.1. MADUREZ

Struts es el framework web J2EE estándar de-facto desde principios del 2000, y ha alcanzado un alto grado de madurez. Tiene una comunidad de desarrolladores muy activa, existe mucha documentación disponible, libros, *mailing-list*, se han escrito centenares de artículos, etc. Las IDEs JAVA más importantes del mercado, lo soportan.

JSF es un framework web J2EE, que ha emergido en el año 2004, cuenta con la ventaja de ser el primer framework J2EE con una especificación que está incluída en la versión actual de J2EE, lo que lo transforma en el primer estándar y de esta manera obliga a todas las implemenetaciones de servidores J2EE a soportarlo. Sin embargo, aún no se conocen resultados masivos de su uso y es necesario esperar un cierto tiempo para analizar su rendimiento en producción en sistemas de mediana envergadura.
Naturalmente, Struts es el más maduro de los dos frameworks, tuvo una gran aceptación desde su aparición, y en la actualidad sigue evolucionando en dos sentidos, Struts 1 y Struts 2.

5.2. DESARROLLO DE PÁGINAS: LA VISTA

Struts cuenta con un conjunto de *custom tags* [Ref. 14] que facilitan la creación de formularios HTML para entrada de datos y que interactúan con los objetos del framework Struts. Provee las mismas funcionalidades que HTML, pero le facilitan al programador la creación de los formularios y la visualización de lo errores. Además, la distribución de Struts tiene integrado el framework **Tiles** [Ref. 15] también de código fuente abierto, que a través de plantillas, extiende las capacidades provistas por Struts para la **Vista**. De esta manera, se mejora el *look&feel* de las aplicaciones.
Por otro lado, el único cliente que soporta Struts, es el navegador de web que despliega páginas HTML dinámicas.
La distribución de JSF, cuenta con un conjunto de componentes de IU básicas, que pueden personalizarse y extenderse creando nuevas componentes de interfaz de usuario con soporte de eventos propios (como por ejemplo tablas *ordenables*, que incluyan imágenes, árboles que representan jerarquías, etc).
Básicamente, la arquitectura de una **componente de GUI JSF** está compuesta por un conjunto de clases JAVA: **Componente IU**, contiene los datos y el comportamiento de la componente del lado del servidor; ***Render***, contiene el código necesario para desplegar la componente IU y traducir las entradas del usuario en valores que entiende la componente; y **clases utilitarias**, estándares o *customizadas* como Conversores, Validadores y *Listeners* de acciones que pueden asociarse a componentes de IU.
Esta arquitectura flexible y extensible, permite por un lado asociar *renders* diferentes para distintas tecnologías clientes, como teléfonos celulares, PADs, entre otros, además del típico cliente web y, por otro construir interfaces de usuario más ricas. Estas características propias de JSF, son imposibles de lograr en Struts.

5.3. FLEXIBILIDAD DEL CONTROLADOR Y MANEJO DE EVENTOS

Struts y JSF implementan el patrón Front-Controller, que centraliza el manejo de peticiones provenientes de clientes. Este objeto está implementado como un servlet *singleton* [Ref.16].
En el caso de Struts, este controlador esta implementado respetando el concepto de caja gris[2], que permite definir puntos de extensión *enchufables* y asi proveer un comportamiento particular, por ejemplo para procesar las peticiones y manejar errores. Teniendo en cuenta que en Struts es posible dividir la aplicación en módulos, podrían coexistir distintos objetos que procesen los requerimientos y los errores de manera particular, enchufados al controlador central.
En el caso de JSF, los puntos extensión tienen una *granularidad* más fina, ya que cada una de las componentes que conforman una página JSF pueden tener asociados comportamientos customizados, entre ellos validaciones, conversiones y procesamientos de eventos.

[2] Si bien el Servlet controlador tiene un comportamiento de caja negra, tiene puntos de extensión que permiten especializarla.

JSF agrega muchos beneficios al controlador único, proveyendo la capacidad de manejar múltiples eventos sobre una página, mientas que Struts puede manejar un único evento por página.

5.4 VALIDACIÓN Y CONVERSIÓN DE DATOS

En Struts la validación se hace validando al objeto **ActionForm** completo, que representa todos los campos del formulario de entrada. En cuanto a la conversión de datos, usa la estándar de JavaBeans.
Por otro lado, JSF permite validar individualmente cada componente del formulario. Se puede validar usando los validadotes estándares, creando métodos validadores en los *backing beans* o creando clases validadoras especiales (validaciones útiles para casos genéricos).
La conversión de datos, también es de granularidad más fina ya que es posible asociarle conversores específicos a las componentes. La distribución estándar de JSF provee conversores de los tipos de datos más comunes como fechas y monedas, además soporta regionalización. De la misma manera que los conversores, es posible crear conversores especiales.

5.5 NAVEGACIÓN

La navegación es una característica clave de Struts y de JSF. Ambos *frameworks* tienen un modelo de navegación declarativo y definen la navegación usando reglas dentro de un archivo de la configuración XML. Existen dos tipos de mecanismos de navegación: **navegación estática,** cuando una página redirecciona directamente a la siguiente y **navegación dinámica**, cuando cierta acción o lógica determina cuál es la siguiente página. JSF y Struts soportan ambos tipos de navegación.
La navegación en **Struts** está basada en objetos **ActionForward**, son quiénes definen "los lugares a donde ir o pasar el control una vez completado el Action". Son los "links" de la aplicación.
La navegación JSF es manejada por objetos *listeners* de eventos, que procesan los eventos generados por las componentes de IU contenidas en las páginas. Estos *listeners* realizan algún procesamiento y luego devuelven un resultado lógico, que es usado por el sistema de navegación para seleccionar la siguiente página a mostrar.
JSF permite definir un control más fino sobre las reglas de navegación a aplicarse en una página. En **Struts** típicamente una petición se corresponde con una acción y una vez finalizada se aplica una regla de navegación. Sin embargo en **JSF** las acciones se codifican por componente y de esta manera es posible que una página que contiene múltiples componentes defina diferentes acciones por cada una de ellas y a su vez compartan la misma regla de navegación.

6. CONCLUSIONES

Desde el punto de vista las aplicaciones que se pueden construir con ambos frameworks, se puede concluir que **JSF** es un framework mucho más flexible y extensible que **Struts**, pero éste no es ningún accidente, Craig McClanahan, creador de **Struts**, lideró la especificación de **JSF**.
JSF está más cerca del verdadero patrón arquitectural MVC, ya que a pesar de trabajar sobre el protocolo sin estado **HTTP** (HyperText Transfer Protocol) mantiene el estado de las componentes de IU en el servidor, posibilitando la creación del aplicaciones orientadas a eventos, con interfaces de alta calidad y usando herramientas para el desarrollo rápido de aplicaciones o RAD (del inglés Rapid Application Development). Mediante la creación capas de abstracción por encima de la API de Servlets, se logró el objetivo de facilitar el desarrollo de aplicaciones web, equiparándolo al desarrollo de una aplicación de escritorio en un lenguaje convencional como VisualBasic o Delphi.
Otra de las características atractivas de **JSF** es que es el primer **framework** J2EE con una especificación incluida en la última versión de J2EE, convirtiéndolo en el primer estándar del mercado. Por este motivo, todos los servidores J2EE deben soportarlo y las principales IDEs incluyen herramientas RAD para crear aplicaciones JSF.

Finalmente podemos agregar, que los temas de este artículo son enseñados en la materia **"Java y Aplicaciones Avanzadas en Internet"**, perteneciente al último año de la carrera Licenciatura en Informática, de la Facultad de Informática de la UNLP, momento en que los estudiantes cuentan con una importante formación en OO (Orientación a Objetos) y Java. En consecuencia, los alumnos adquieren criterios para evaluar los distintos **frameworks** de código abierto, que muchos de ellos utilizarán a un futuro cercano.

7. REFERENCIAS

[Ref.1] *Java Platform, Enterprise Edition (Java EE),* http://java.sun.com/javaee/index.jsp.
[Ref.2] *Apache Struts*, http://struts.apache.org/
[Ref.3] *Struts in Action; Building web applications with the leading Java framework.* Ted N. Husted, Cedric Dumoulin, George Franciscus, David Winterfeldt; ISBN: 1930110502; Manning Publications.
[Ref.4] *JavaServer Faces*, http://struts.apache.org/
[Ref.5] *Servlets and JavaServer Pages – The J2EE Technology Web Tier* . Jayson Falkner, Kevin Jones; ISBN: 0321136497; Addison-Wesley.
[Ref.6] *JavaBeans Specification,* http://java.sun.com/products/javabeans/docs/spec.html
[Ref.7] *The Apache Velocity Project*, http://velocity.apache.org/
[Ref.8] *XSL Specification,* http://www.w3.org/TR/2006/REC-xsl11-20061205/
[Ref.9] *Java Community Process*, http://jcp.org/en/home/index
[Ref.10] *Especificación de JSF 1.2*, JSR 252, http://jcp.org/en/jsr/detail?id=252
[Ref.11] *JavaServer Faces in Action*. Kito D. Mann; ISBN: 1932394117; Manning Publications.
[Ref.12] *Catálogo de patrones J2EE*,
http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html
[Ref.13] Struts 2, http://struts.apache.org/download.cgi#struts206
[Ref.14] *HTML Taglib* Guides; http://struts.apache.org/1.x/struts-taglib/dev_html.html
[Ref.15] *Tiles Guide*; http://struts.apache.org/1.x/struts-tiles/dev_tiles.html
[Ref.16] *Design Patterns, Elements of Reusable Object-Oriented Software*; Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; ISBN 0-201-63361-2.