Diseño e implementación de un marco de trabajo de presentación para aplicación J2EE

Felipe Benavente Cabrera

Ingeniería en Informática

Josep Maria Camps Riba

UOC - 20 de junio de 2011

Justificación

- J2EE es posiblemente el estándar de desarrollo más usado.
- En estos entornos, gran parte del tiempo es para el desarrollo de la capa de presentación:
  - La interfaz de usuario es un componente complejo.
  - Código fuente disperso: JSP, Servlets, Beans ...
- Una buena solución puede ser disponer de un marco de trabajo con las siguientes características:
  - Estructure el código en componentes independientes: Modelo, Vista, Controlador (**Patrón MVC**).
  - Disponga de un conjunto de utilidades que agilicen el desarrollo.

Objetivos

Para el desarrollo del marco de trabajo nos fijaremos los siguientes objetivos:

- Estudiar y evaluar marcos de trabajo de presentación existentes en el mercado.
- Diseñar e implementar un marco de presentación.
- Implementar una pequeña aplicación que muestre su funcionamiento.

Planificación

Para el cumplimiento de los objetivos se fijan los siguientes hitos:

- **8 de abril**, entrega del estudio y evaluación de marcos de trabajo de presentación actuales así como el diseño.
- **29 de abril**, entrega de la implementación del marco de trabajo base y una aplicación de pruebas.
- **17 de mayo**, entrega de la implementación mejorada del marco y la aplicación de pruebas adaptada.
- **20 de junio**, entrega de la memoria y presentación del proyecto.

Como suele ser habitual, esta planificación sufrirá cambios

Aspectos técnicos

Antes de empezar a trabajar en un proyecto es necesario hacer un estudio de los aspectos que le atañen, en nuestro caso:

- Patrones de diseño. El patrón MVC.
- Marcos de trabajo (***Frameworks***)
- *Frameworks* de presentación
  - Struts
  - Spring MVC
  - Java Server Faces (JSF)

Patrones de diseño. El patrón MVC

Los patrones de diseño son plantillas que ofrecen soluciones genéricas a problemas comunes.

Uno de los patrones más conocidos es el patrón MVC, gracias al cual se consigue aislar la interfaz gráfica de la lógica de la lógica de negocio. Para ello se divide la lógica de presentación en 3 componentes que dan nombre al patrón:

- Model (M), encapsula la lógica de negocio.
- View (V), presenta la información al usuario.
- Controller (C), gestiona los eventos producidos por el usuario interactuando con el modelo y presentando las vistas adecuadas.

Frameworks

Los *frameworks*, o marcos de trabajo, son un conjunto de librerías, clases u otros ficheros que definen y/o implementan un comportamiento genérico y que suelen incluir un conjunto de utilidades para agilizar el trabajo del programador.

En el caso que aplica al proyecto nos centraremos en los marcos de trabajo de presentación que además implementan el patrón de diseño MVC.

Nos centramos en tres de los más importantes:

- Struts, en sus dos versiones (1 y 2).
- Spring MVC.
- Java Server Faces (JSF).

Frameworks - Struts

- Struts 1, uno de los primeros marcos de presentación y de los que tienen una vida más larga. Se compone de:
  - Controlador (ActionServlet), configurable mediante un fichero (struts-config.xml) y se encarga de dar respuesta a las peticiones del cliente.
  - Action, contienen la lógica de la capa de presentación.
  - ActionForms, contienen los datos necesarios para la capa de presentación.
  - A parte contiene utilidades de validación y conversión para los datos enviados a los formularios, otras utilidades para la internacionalización y un conjunto de *taglibs* que ayudan al desarrollo de las vistas.
- Struts 2, no es propiamente una evolución del primero, ya

Frameworks - Spring MVC

Forma parte de un marco más potente para el desarrollo de aplicaciones (Spring).

- El DispatcherServlet es el controlador principal, configurable por ficheros XML o anotaciones.
- La lógica de la capa de presentación se encuentra en los llamados Controllers, más simples de implementar que los de Struts 1.
- Permite diferentes tipos de vistas. No sólo JSP.
- Al igual que el anterior contiene utilidades de validación, conversión, internacionalización y *taglibs*.

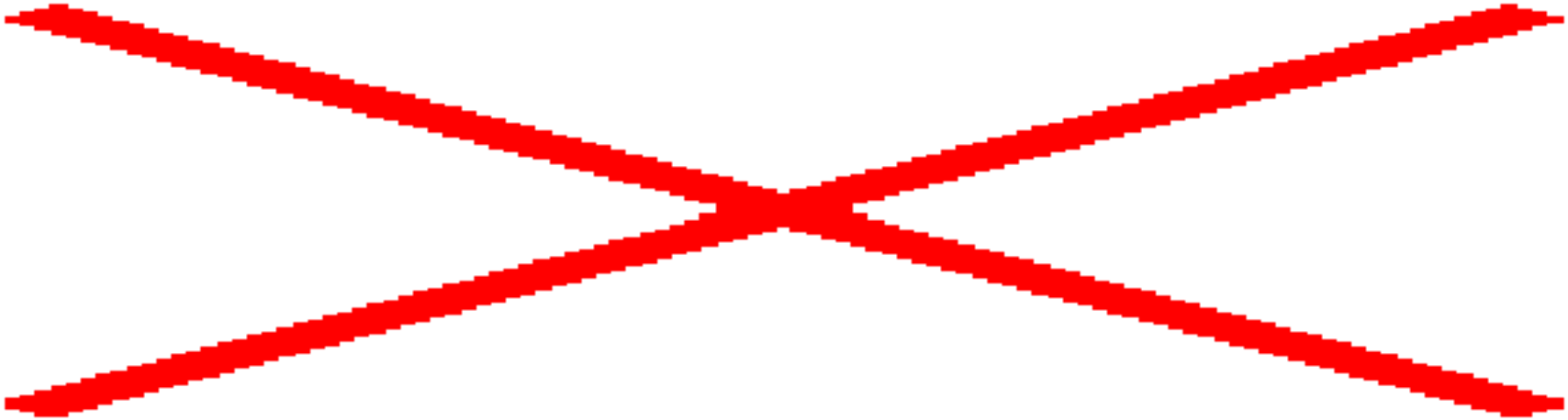
Frameworks - Java Server Faces

Es una especificación propia de JEE, por lo que permite diferentes implementaciones.

- Basado en componentes, como si de una aplicación de escritorio se tratara.
- También contiene un controlador principal (FacesServlet) configurable mediante ficheros XML y anotaciones.
- La lógica se encuentra en los ManagedBeans.

Frameworks – tabla comparativa

Del estudio anterior obtenemos la siguiente tabla comparativa



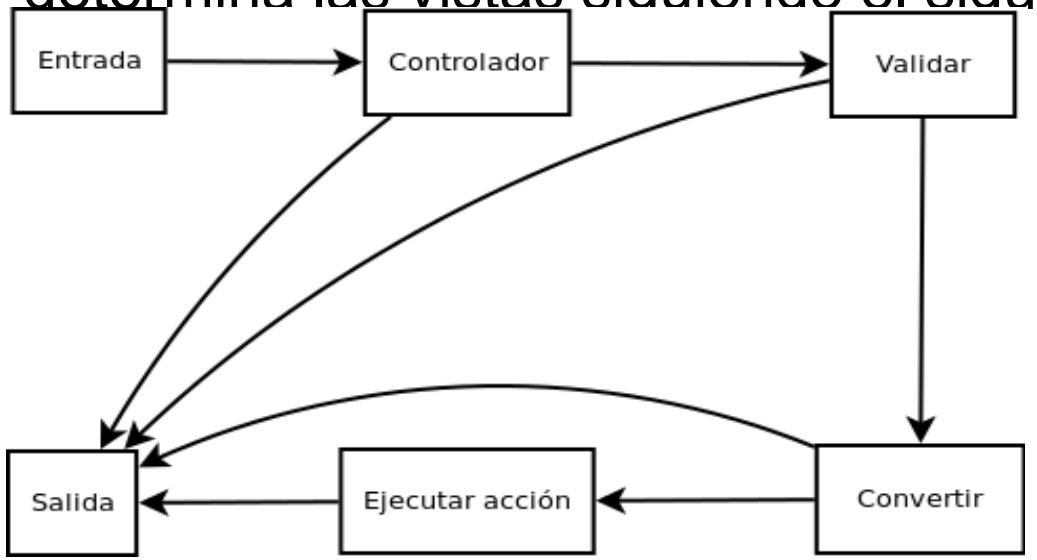
Diseño

Partiendo de la información anterior, parece claro que el marco debe tener las siguientes características:

- Un controlador principal encargado de gestionar las peticiones.
- Un sistema de configuración para determinar el comportamiento del controlador y definir los componentes básicos.
- Utilidades de validación y conversión de datos.
- Debe permitir la internacionalización.
- Permitir el uso de taglibs estándares.
- Interfaces que ayuden al usuario a definir los componentes básicos, aunque es preferible que permita componentes de

Diseño – Controlador principal

El controlador es el componente principal. Gestiona las peticiones, valida y convierte los datos, interactúa con el modelo y determina las vistas siguiendo el siguiente flujo:



Dicho controlador será configurado mediante un fichero XML.

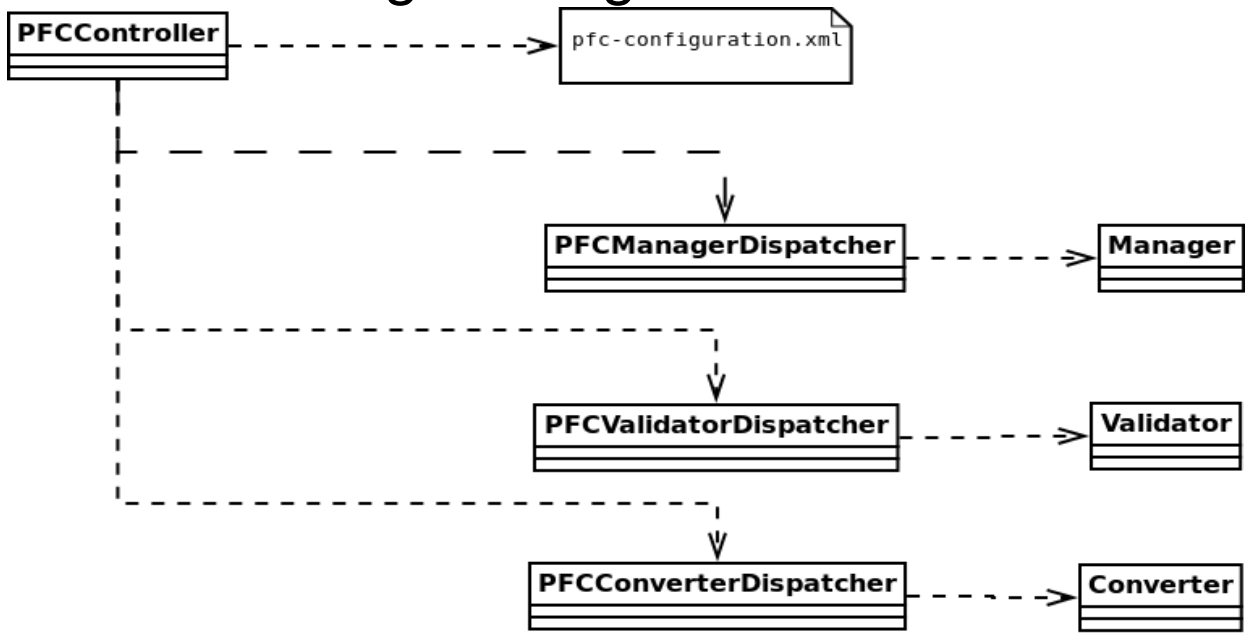
Diseño – Componentes

El controlador delegará cada una de las acciones en los siguientes componentes:

- PFCManager, serán los componentes que el programador deberá desarrollar y contendrán la lógica y datos de presentación.
- PFCValidator, encargados de validar los datos introducidos por el usuario. Serán proporcionados por el marco aunque podrán ser ampliados por el usuario.
- PFCConversor, convierten los datos introducidos por el usuario a objetos que los managers pueden tratar. También son proporcionados por el marco y pueden ser ampliados.

Diseño – Delegación de funciones

Para facilitar la programación del controlador se ha decidido delegar la gestión de los componentes en disparadores, tal como muestra el siguiente gráfico:



Implementación – Entorno de desarrollo

El primer paso para proceder con la implementación ha sido preparar el entorno de desarrollo que en nuestro caso ha sido el siguiente:

- Sistema Operativo: Ubuntu 11.04.
- IDE: Netbeans 6.9.
- Versión java: 1.6.
- Servidor: Apache Tomcat 6.0 sobre Ubuntu 11.04.

Implementación – Configuración

Para la configuración del controlador hemos decidido usar un fichero XML y transformarlo a objetos java para tratarlo desde el controlador. Para ello:

- Se ha definido el contenido del XML mediante un fichero DTD.
- Se ha utilizado la librería XStream para pasar el contenido del XML a una serie de objetos.
- Se ha implementado una clase que se lanza durante el inicio de la aplicación que será la encargada de ejecutar la lectura y registrarla en un objeto accesible dentro del contexto de la aplicación.

Implementación – Controlador

Siguiendo el flujo definido en el diseño el controlador sigue los siguientes pasos:

- Extraer de la petición el mapeo adecuado. De él se obtienen, las validaciones, los conversores, el manager y método del manager, así como las rutas de salida en función del resultado.
- Validar los datos de la petición delegando en el PFCValidatorDispatcher.
- Convertir los datos a valores del manager o parámetros del método a ejecutar delegando en PFCConverterDispatcher.
- Ejecutar el método del manager apropiado a través del PFCManagerDispatcher.
- Si todo es correcto se muestra la visa configurada por el

Aplicación de pruebas

La aplicación de pruebas es un carro de la compra sencillo con las siguientes funciones:

- Pantalla de selección de productos.
- Consulta y modificación del carro.
- Confirmación de la compra realizada.
- Introducción de los datos del cliente.

Conclusiones

- La capa de presentación es una de las partes más costosas en el desarrollo de aplicaciones J2EE.
- Usar el patrón de diseño MVC permite tener componentes independientes donde cada uno tiene unas responsabilidades concretas.
- Usar marcos de trabajo que implementen el patrón MVC permiten al programador centrarse exclusivamente en la programación de la aplicación.
- Usando marcos establecidos en el mercado nos aprovecharemos de la experiencia de éstos y darán una garantía a nuestro programa.

Aspectos a mejorar

Por diversas causas, el desarrollo del marco no ha sido tal cual se planificó y se ha quedado en un sistema básico, por lo que carece de ciertas funciones que pueden resultar interesantes:

- Configuración del marco mediante anotaciones.
- Integración con AJAX.
- Subida de ficheros.

Además, puntos detallados en diseño que no se han podido alcanzar como:

- Mayor número de conversores y validadores.
- Obtención de parámetros a través de la URL de la petición.