Diseño e implementación de un framework de presentación

Enrique Mengíbar Vázquez
Ingeniería de Informática, 2.º ciclo

Director: Óscar Escudero Sánchez
Universitat Oberta de Catalunya
Barcelona, enero de 2012

Justificación

Problema

- La capa de presentación es una de las partes más importantes en una aplicación web
  - Mezcla de la lógica de la aplicación con la presentación
  - Difícil de mantener y compleja

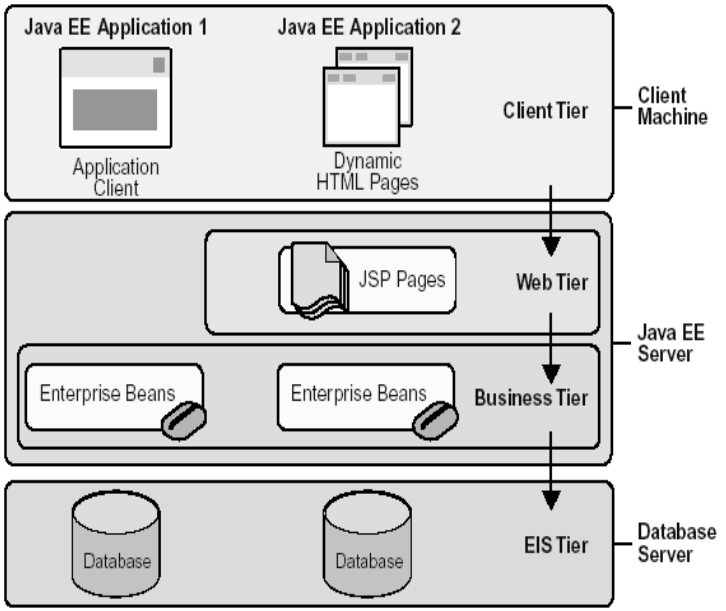
Solución

- Plataforma J2EE fuertemente consolidada en el mercado
- Incorporación de los patrones de diseño a la Ingeniería del Software. Catálogo *Core J2EE Patterns de Sun*.
- Patrón MVC la separación en módulos proporciona separación de los datos, interfaz y lógica. Modificación de la interfaz impacto mínimo
- Frameworks de desarrollo como conjunto de librerías y componentes utilizados para implementar la estructura estándar de una aplicación

Plataforma J2EE

- J2EE colección de especificaciones que ofrecen un marco de trabajo común y una serie de convenciones, junto un conjunto de servicios sobre los cuales desarrollar aplicaciones multicapa
- Permiten al desarrollador centrarse en el diseño e implementación del sistema, delegando las tareas típicas y problemas de bajo nivel ajenas a la propia aplicación a la infraestructura del servidor de aplicaciones

- **Capa Cliente.** Interfaz gráfica del sistema encargada de interactuar con el usuario. J2EE soporta diferentes tipos de clientes incluyendo clientes HTML, applets Java y aplicaciones Java
- **Capa Web**. Ubicada en el servidor Web y contiene la lógica de presentación. Recibe los datos del usuario desde la capa cliente y genera una respuesta apropiada a la solicitud. J2EE utiliza aquí los componentes Servlets y JSP
- **Capa Negocio**. Se encuentra en el servidor de aplicaciones y contiene el núcleo de la lógica del negocio de la aplicación. Provee las interfaces necesarias para utilizar el servicio de componentes del negocio
- **Capa EIS**. Responsable del sistema de información empresarial o Enterprise Information System (EIS)

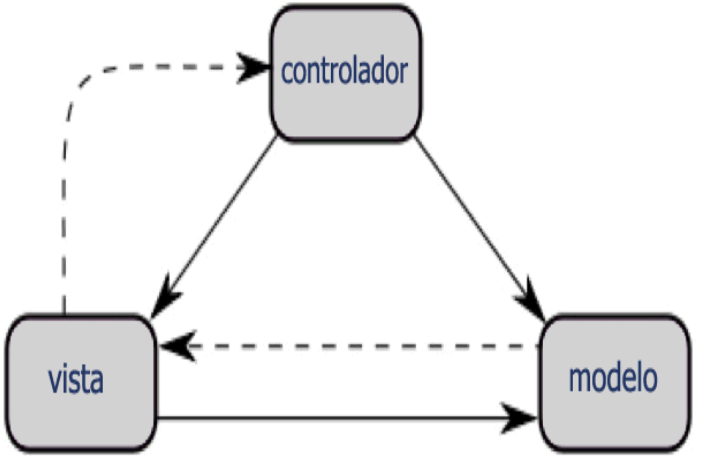


Patrones

- Un patrón es una solución a un problema no trivial de diseño que es
  - **Efectiva**: ya se resolvió el problema satisfactoriamente en ocasiones anteriores
  - **Reusable**: se puede aplicar a diferentes problemas de diseño en distintas circunstancias.
- Una solución se considera un patrón si posee ciertos componentes:
  - **Nombre del patrón**. Permite describir brevemente un problema de diseño junto con sus soluciones y consecuencias
  - **Problema**. ¿Qué hace?, ¿Cuál es su razón y propósito?, ¿Qué cuestión de diseño particular o problema aborda?
  - **Solución**. Escenario que ilustra un problema particular y cómo el patrón lo resuelve.
  - **Consecuencias**. Resultados en términos de ventajas e inconvenientes.
- *Core J2EE Patterns* presenta un catálogo completo de patrones de diseño y arquitéctónicos específicos para los problemas comunes en J2EE. También se identifican malas prácticas que deben evitarse. Está organizado en:
  - Patrones de la **capa de presentación** que describen las soluciones que implican JSP y servlets
  - Patrones en la **capa de negocio** que expone soluciones relacionadas con Enterprise Java Bean (EJB)
  - Patrones de la **capa de integración** describe soluciones que implican JDBC y el servicio Java Message Service (JMS)

Arquitectura MVC

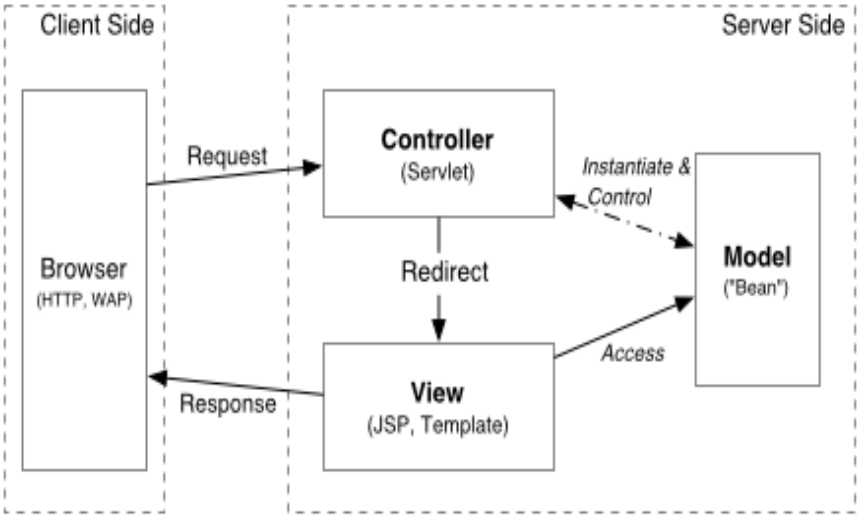
- Las aplicaciones Web en general tienen tres aspectos a considerar en su desarrollo:
  - El código de acceso, inserción, consulta actualización y/o eliminación de los datos
  - El código del diseño de las páginas a mostrar
  - El código que controla el flujo de las páginas
- En la práctica se hace patente que si una aplicación Web empresarial tiene estos aspectos muy mezclados se vuelve difícil de mantener



- Una forma de separar estos aspectos es usando la arquitectura Model-View-Controller (MVC). Se trata de una arquitectura de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones en un componente tienen un mínimo impacto en el resto
  - El **modelo** representa a la lógica de negocios (manipulación de datos).
  - La **vista** representa la presentación de los datos (diseño de páginas).
  - El **controlador** representa el código de navegación de la aplicación (control de flujo)

Arquitectura MVC – Model 2

- **Model 2** es la variación de la arquitectura MVC, recomendada para las aplicaciones web desarrolladas sobre J2EE
- Consiste en el desarrollo de una aplicación según el patrón de diseño MVC, pero especificando que el controlador debe estar formado por un único **servlet**, que centralice el control de todas las peticiones al sistema, y que basándose en la URL de la petición HTTP y en el estado actual del sistema, derive la gestión y control de la petición a una determinada **acción** de entre las que se encuentren registradas en la capa controlador



- Una ventaja de MVC Model 2 es la escalabilidad, ya que al tener componentes separados, es sencillo añadir más componentes donde sea necesario.
- Otra ventaja es su capacidad para gestionar en un único punto la aplicación de filtros a las peticiones, las comprobaciones de seguridad, el registro de logs, etc

Frameworks de desarrollo

- Un framework es un conjunto de librerías y componentes que implementan la estructura de una aplicación que junto con una documentación y metodología de uso nos permite diseñar, construir e implantar aplicaciones empresariales de forma uniforme, rápida y de mayor calidad
- Determinan la arquitectura de una aplicación, su estructura general, sus particiones en clases y objetos, las responsabilidades clave, así como la colaboración entre dichas clases y objetos. Todos esto es definido por el framework, evitando que lo haga el usuario, permitiendo que el enfoque pueda hacerse sobre la propia aplicación en lugar de hacerlo sobre su arquitectura
- El framework captura las decisiones de diseño que son comunes a su dominio de aplicación. Un framework no sólo promueve la **reutilización de código** sino también la **reutilización de diseño**.
- Los principales frameworks para el desarrollo Web bajo J2EE son
  - Apache Struts
  - Apache Struts 2.
  - Spring Framework.
  - Java Server Faces.

Struts

- **Apache Struts** es un framework de software libre orientado al desarrollo de aplicaciones web basadas en J2EE, ajustándose al patrón MVC
- Provee su propio componente Controlador Web y se integra con otras tecnologías para proveer el Modelo y la Vista
  - Para el **Modelo**, puede interactuar con tecnologías de acceso de datos estándares como JDBC y EJB, así como también muchos paquetes de terceros, como Hibernate, iBATIS.
  - Para la **Vista** trabaja bien con JavaServer Pages incluyendo JSTL y JSF y otros sistemas de presentación.
  - El **Controlador** actúa como un puente entre el Modelo y la Vista Web.
- Proporciona las siguientes características
  - Un conjunto de etiquetas personalizadas JSP para su uso en la capa view de MVC
  - Varias opciones para la validación de entrada de usuario en formularios HTML
  - Mecanismos para el manejo e informe de errores
  - Soporte para la internacionalización (i18n) a través de ficheros de recursos y Java Locales
  - Soporte para fuentes de datos

Struts 2

- **Struts2** es una evolución de Struts con el objetivo de simplificar aún mas el desarrollo de la capa de presentación de las aplicaciones web. Se trata de una fusión con el framework Webwork2.
- Las características principales de Struts2 y mejoras respecto a la versión anterior son:
  - Orientado a acciones
  - Arquitectura flexible mediante plugins e interceptores que permiten personalizar el tratamiento de las peticiones hasta llegar a cada acción de forma individual o para un conjunto de acciones
  - Proporciona un sistema de validación flexible
  - Permite utilizar cualquier clase Java normal (POJO) como acción
  - Conversión de tipos automática que mapea de forma transparente los valores HTTP a las acciones
  - Motor integrado de inyección de dependencias que permite inyectar componentes
  - Proporciona anotaciones Java como alternativa a la configuración declarativa
  - Gran cantidad de etiquetas, calendarios para seleccionar fechas, vistas en árbol, etc.
- Struts2 es un framework MVC model 2 con la particularidad de que se permite que las acciones tomen el rol del modelo. Esto significa que las vistas pueden obtener datos directamente de la acción sin que sea necesario disponer de un objeto con datos intermedio disponible para enviar desde el modelo a la vista.

Spring

- **Spring** es un framework ligero basado en la técnica **Inversión de Control** (IoC) y una implementación de desarrollo según el paradigma de la **Programación Orientación a Aspectos** (AOP).
- No obliga a usar un modelo de programación concreto. Es un framework modular que cuenta con una arquitectura divida en siete capas o módulos. Se permite utilizar solamente los módulos que nos interesen.
  - **Sprint Core**. Proporciona la funcionalidad esencial del framework, está compuesta por el BeanFactory, el cual utiliza el patrón de *Inversión de Control*, los objetos se configuran a través de *Inyección de Dependencia*
  - **Spring Contex**t. Aarchivo de configuración que proporciona información contextual al framework
  - **Spring AOP**. La *Programación Orientada a Aspectos*, es un modelo de programación que aborda un problema específico: capturar las partes de un sistema que los modelos de programación habituales obligan a que estén repartidos a lo largo de distintos módulos del sistema
  - **Spring ORM**. Spring no dispone de un módulo ORM (*Object-Relational Mapping*) propio. El módulo ORM soporta los frameworks ORM más populares del mercado, como son Hibernate, iBATIS, etc.
  - **Spring DAO**. Soporte para el conocido patrón DAO
  - **Spring Web MVC**. Spring incluye un framework para el desarrollo web basado en el patrón MVC. Aunque es similar en algunas características a otros frameworks populares, Spring Web MVC posee algunas ventajas que le diferencias de otros frameworks

Java Server Faces (JSF)

- **JSF** es un framework para aplicaciones Web en Java de Sun Microsystems
- Está **orientado a la interfaz gráfica de usuario** (GUI) facilitando su desarrollo, sin embargo realiza una separación eficaz entre componentes y presentación
- Brinda un modelo basado en componentes y dirigido por eventos para el desarrollo de aplicaciones web, que es similar a **modelo usado en aplicaciones GUI** de escritorio tradicionales
- Se trata de una **especificación** de la que existen distintas implementaciones: *JSF Reference Implementatio* de *Sun Microsystems*, *MyFaces de Apache Software Foundation*, etc.
- Los principales componentes de la tecnología JSF son
  - Un API y una implementación de referencia para: representar componentes UI y manejar su estado; manejo de eventos, validación en el lado del servidor y conversión de datos; definición de navegación entre páginas; soporte a la internacionalización y accesibilidad.
  - Una librería de etiquetas JSP personalizada para representar componentes dentro de una página JSP

Análisis del framework

Requisitos

- **Control declarativo** del flujo de navegación mediante reglas de forma externa al código
- Se proporcionan mecanismos para facilitar el proceso de **validación y conversión** de los datos de entrada
- **Gestión de errores.** El framework proporciona excepciones para el manejo de errores y mecanismos que automatizan la presentación del mensaje de error al usuario
- **Internacionalización**. Nos permitirá mantener distintos juegos de mensajes para ser usados en la interface de la aplicación en función del idioma.
- Para ayuda en el soporte a la Internacionalización se ha diseñado una **librería de etiquetas especializadas** que permite insertar mensajes en la vista que serán posteriormente adaptados al idioma seleccionado

Patrón Service to Worker

Service to Worker

- Define un marco global donde se separan las distintas responsabilidades
- Se organiza alrededor de otros patrones. El control centralizado es realizado por un **Front Contoller** y el manejo de las peticiones y creación de vistas es realizado por un **Application Controller**

Funcionamiento

- Front Controller recibe la petición y la delega al Application Controller
- El Application Controller resuelve la petición entrante al comando apropiado para invocarlo posteriormente
- El comando invocará a la capa de negocio y devolverá un modelo apropiado para ser presentado en la vista

Ventajas

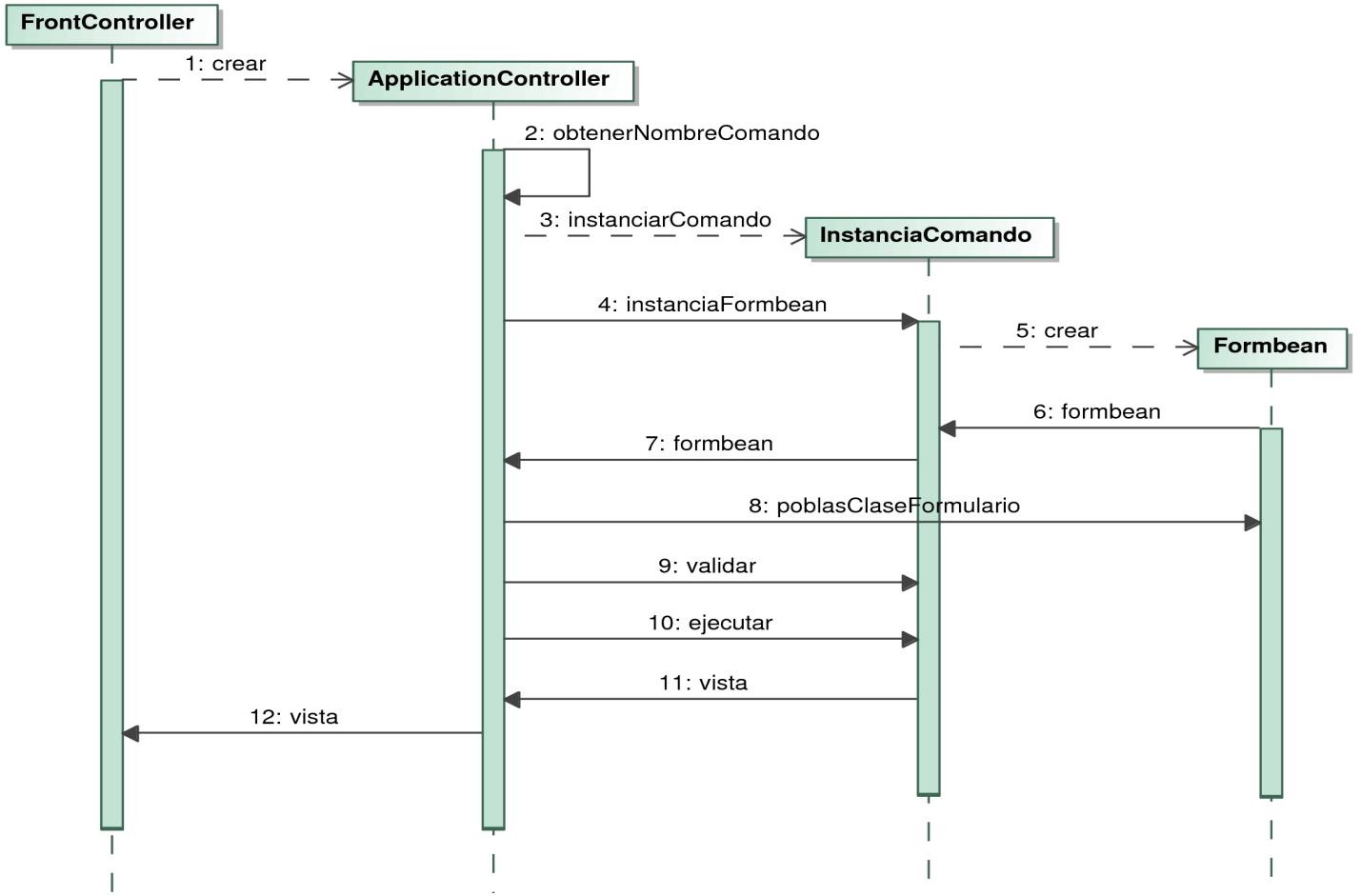
- Se centraliza el control y se logra un sistema mas modular, reusable y mantenible.
- Un controlador como punto inicial para la gestión de peticiones permite realizar algunas funciones como: comprobación de restricciones de seguridad, manejo de errores, mapeo y delegación de peticiones a otros componentes

Diseño del framework

- Un **Comando** es cada una de las acciones que se ejecutan en el servidor asociadas al procesamiento o submit de un de un formulario HTML
- Un **Formbean** es una clase que facilita el acceso a los campos de entrada del formulario HTML desde los objetos comando
- La **Configuración** se realiza de forma declarativa, se carga desde un fichero xml en el método *init* del *servlet.* Serealiza un parseo dirigido por ejeventos mediante el el componente *Digester* de Apache
- La **Validación** de los datos se realiza en el servidor en la propia clase comando, las reglas se deben implementar programáticamente en Java.
- Los **Errores** son propagados en la pila de llamadas hacia arriba y capturados por el Application Controller que se encarga de redireccionar a una vista genérica que muestra el mensaje al usuario
- **La internacionalización (i18n)** se define mediante la configuración de distintos ficheros de mensajes *properties* en diferentes idiomas que serán utilizados en función del *locale* del navegador o de un valor *locale* global establecido en la configuracion

Manejo de las peticiones

El diagrama de secuencia representa el intercambio de mensajes entre el Front Controller, el Application Controller y el resto de las clases involucradas



Mejoras

- **Validación en el lado del cliente** mediante Javascript. Detectar datos mal formados en el propio cliente evita tener que realizar una llamada al servidor, aportando agilidad a la aplicación
- **Librerías de etiquetas avanzadas** que nos permitan definir nuevas acciones propietarias que nos permitan evitar código repetitivo y simplificar el desarrollo de las vistas
- EL uso del patrón **Service to Worker** nos abstraer del mecanismo petición-manejo propio de la tecnología Web. Esta mejora de modularidad nos permitiría incorporar fácilmente mecanismos como **autenticación** y **control de acceso**, **trazas de actividad**,etc.
- La posibilidad de contar con la generación de **formularios dinámicos** definidos declarativamente nos evitaría tener que definir un Formbean por cada Comando.
- Proporcionar soporte para **AJAX** que nos permitiera realizar peticiones asíncronas al servidor de aplicaciones. Esto nos permitiría lograr que la capa de presentación fuera más rápida, ágil y dinámica.

Conclusiones

- En la plataforma J2EE, el desarrollo de la **capa de presentación** es costoso, las malas prácticas conducen a la **mezcla entre la lógica de la aplicación y la presentación**
- Los **Patrones de diseño** han supuesto un cambio importante en la forma de afrontar el desarrollo, nos aportan **soluciones efectivas** (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y **reusables** (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).
- El uso de **frameworks** para el desarrollo no sólo promueve la **reutilización de código** sino también la **reutilización de diseños y arquitecturas** consolidadas.
- La experiencia adquirida durante el desarrollo del presente proyecto fin de carrera ha permitido
  - Disponer de una visión de las diferentes soluciones y frameworks existentes en el mercado
  - Conocer las últimas tendencias en cuanto a arquitectura y diseño de software empresarial
  - Adquirir práctica en el desarrollo Java dentro de la plataforma J2EE