Proyecto de fin de Carrera
2013/2014

Diseño e implementación de un framework de presentación: El framework Myst

Autor: Miguel Souto Bartolomé
Consultor: Óscar Escudero Sánchez

Índice:

1. Introducción

1.1 Resumen ejecutivo

Aplicaciones empresariales

- ✓ Millones de usuarios
- ✓ Usadas masivamente en Internet
- ✓ Grandes expectativas comerciales

- x Complejas de programar
- x Conocimientos especiales

Soluciones

Patrones

- Soluciones contrastadas y de calidad
- Máxima divulgación

Frameworks

- Combinación de soluciones
- Amplia variedad
- Generales y especializados

1. Introducción

1.2 Objetivos

Estudio tecnológico

- Patrones
  - Capa de presentación
  - M-V-C
- Frameworks
  - Elección de cuatro frameworks
  - Extracción de características
  - Estudio comparativo

Creación de un framework JEE

- Capa de presentación
- Uso de patrones
- Funcionalidad extraída del estudio
- Aplicación de muestra

Myst framework

2. Patrones

Ventajas

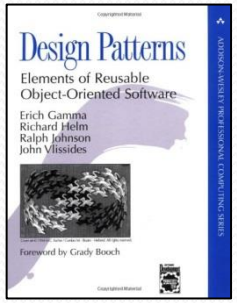
- ✓ Repositorio de conocimientos
- ✓ Soluciones útiles contrastadas
- ✓ Ahorran tiempo y esfuerzo

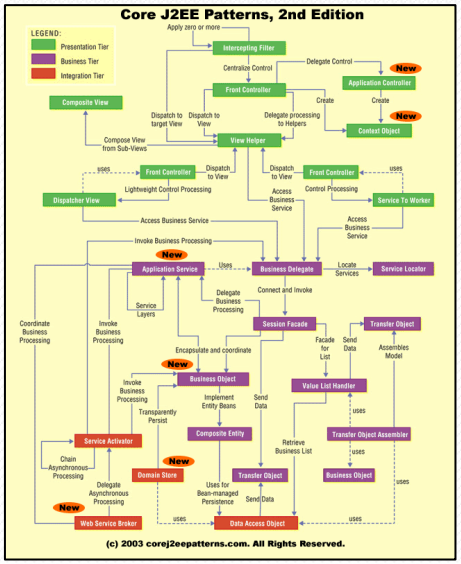
críticas

x ¿Carencias propias de lenguajes OO?

Variedades de interés

- Gang of Four
- Patrones web
- Patrones JEE





2. Patrones

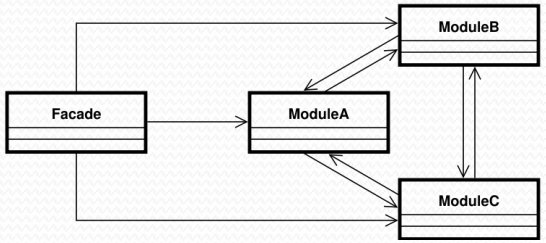
2.1 Patrones de aplicaciones web (I)

Singleton

- Una clase, una instancia

Fachada

- Interfaz única para acceder a un subsistema

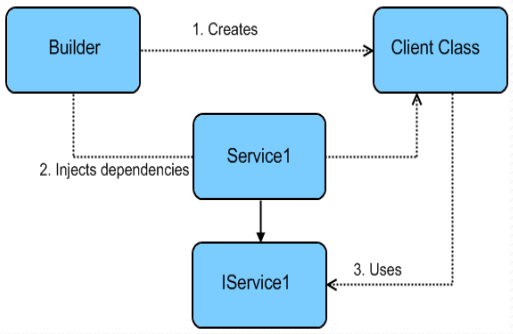


Command

- Una interfaz, uno o más métodos
- Facilita la parametrización
- Se delega el comando concreto a la ejecución

Inyección de dependencias

- Un constructor suministra objetos a una clase
- Permite seleccionar implementaciones en ejecución

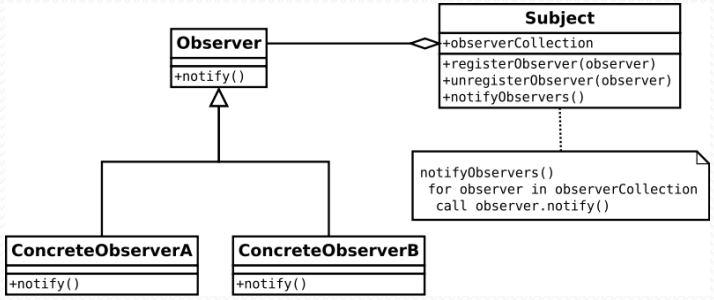


2. Patrones

2.1 Patrones de aplicaciones web (y II)

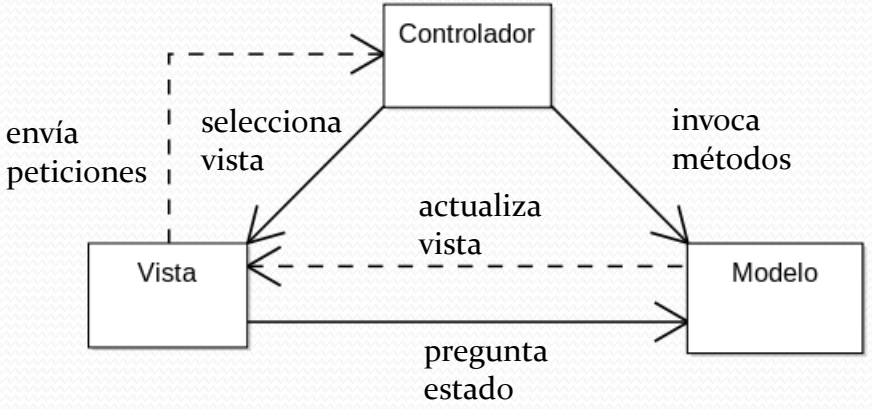
Observador

- Objeto abstracto vigila y notifica cambios en objetos dependientes
- Base del controlador del MVC



Modelo-Vista-Controlador

- **Modelo**: datos de la aplicación y lógica del negocio
- **Vista**: presenta la información al usuario
- **Controlador:**
  - Recibe peticiones, y las convierte en órdenes al Modelo
  - Actualiza la Vista

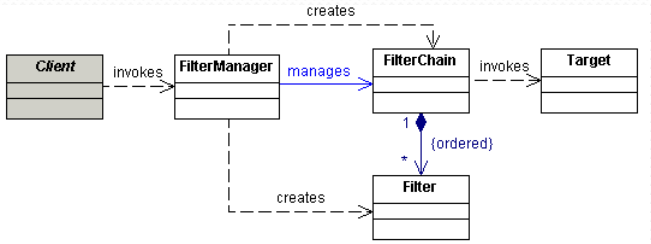


2. Patrones

2.2 Patrones JEE basados en MVC

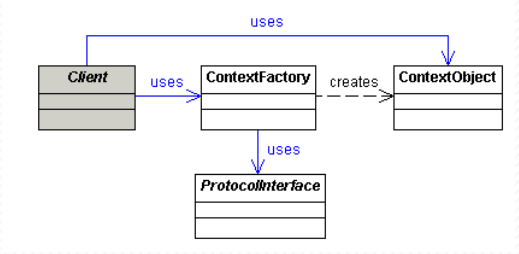
Intercepting filter

- Se interceptan las peticiones
- Tareas previas y posteriores a la petición



Context Object

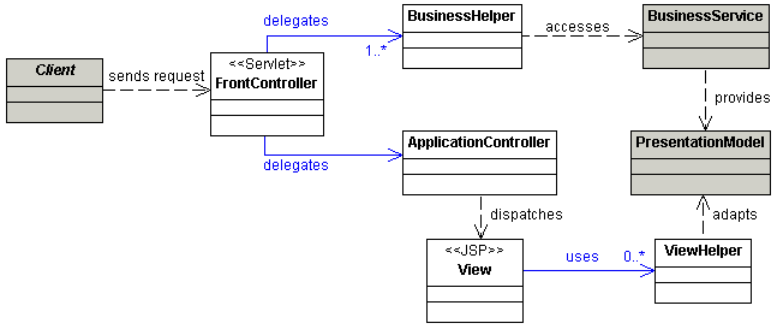
- Clase especial, guarda el estado del sistema
- Accesible por toda la aplicación



Service to Worker

- Centraliza la gestión de las peticiones
- Gestiona la lógica de negocio y las vistas
- Combinación de patrones:

  Front Controller, Application Controller, View Helper, Dispatcher View



3. Frameworks

Ventajas

- ✓ Abstracción sobre la problemática del servlet
- ✓ Bibliotecas, APIs, compiladores y herramientas de ayuda
- ✓ Facilita la programación
- ✓ Reduce tiempo y esfuerzo

Inconvenientes

- x Requieren de un aprendizaje previo
- x Cada framework es diferente
- x La bibliografía en ocasiones es escasa

Frameworks de presentación

- ✓ Gestión de HTML
- ✓ Gestión del Servlet
- ✓ Control de la navegación
- ✓ Generación de vistas

¿ Criterio de selección para el estudio ?

Popularidad

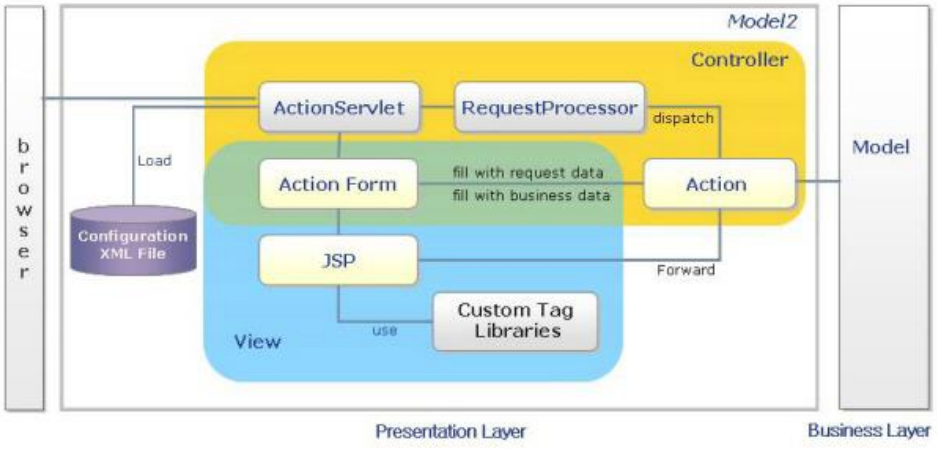
3. Frameworks

3.1 Struts

- Orientación a acciones y URLs

- ✓ Abstracción parcial de la complejidad del servlet
- ✓ Validación de datos con los formularios Actionform
- ✓ Potente colección de tags
- ✓ Varias tecnologías para la Vista
- ✓ Amplia bibliografía

- x Action implementa el patrón Singleton
- x Actions y Actionforms no son POJO
- x Aplicaciones fuertemente acopladas
- x Recientemente abandonado
- x Configuración en XML



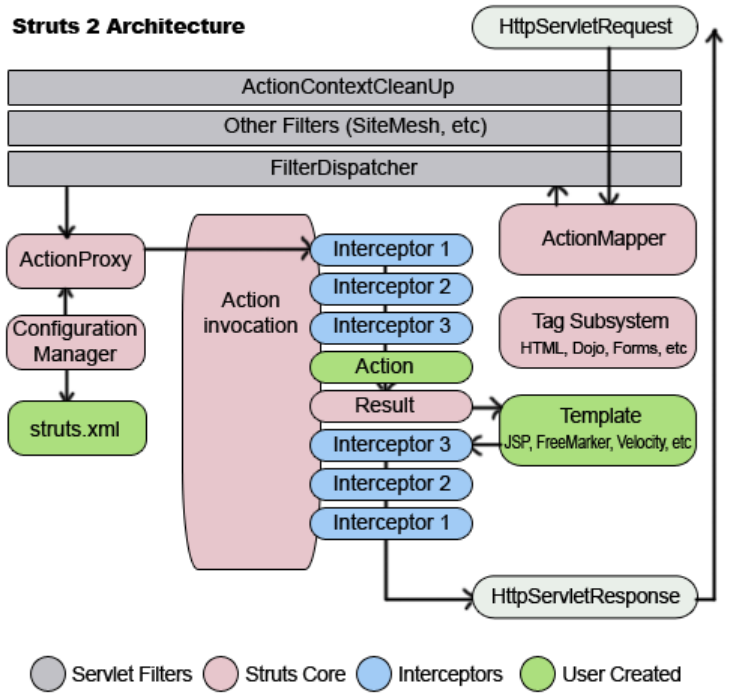
3. Frameworks

3.2 Struts^2

- Orientación a acciones y URLs

- ✓ Todas las virtudes de Struts
- ✓ Actions y Actionforms son objetos POJO: menos acoplamiento
- ✓ Action no implementa Singleton
- ✓ Tags mejorados
- ✓ Compatible con AJAX
- ✓ Inyección de dependencias en las Actions a través de motores externos
- ✓ Implementa filtros e interceptores

- x Configuración en XML
- x Filtros e interceptores aportan la misma funcionalidad

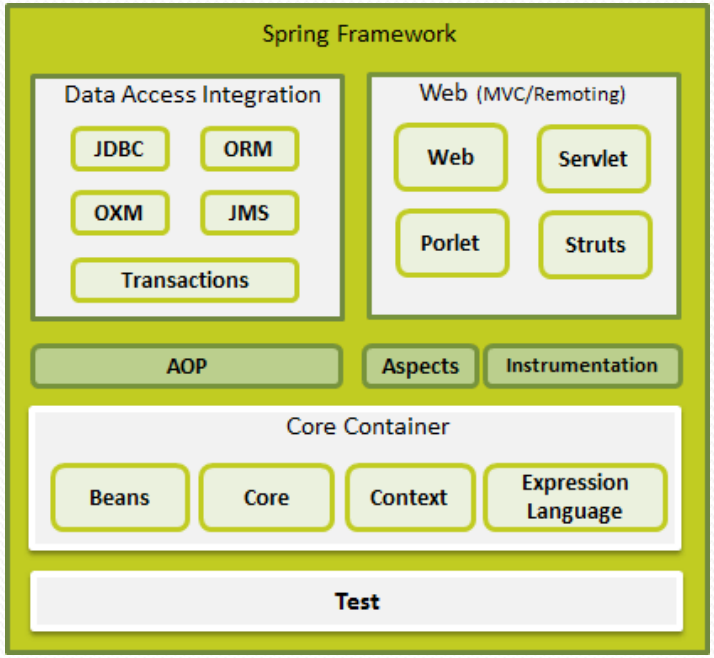


3. Frameworks

3.3 Spring

- Orientación a acciones y URLs

- ✓ Framework de pila completa
- ✓ Utilización de objetos POJO
- ✓ Potente motor de inyección, uso de inversión de control
- ✓ Permite programación orientada a aspectos
- ✓ Convención sobre configuración
- ✓ Populares capas de negocio y acceso a datos
- ✓ Facilita la realización de pruebas
- ✓ Amplia bibliografía



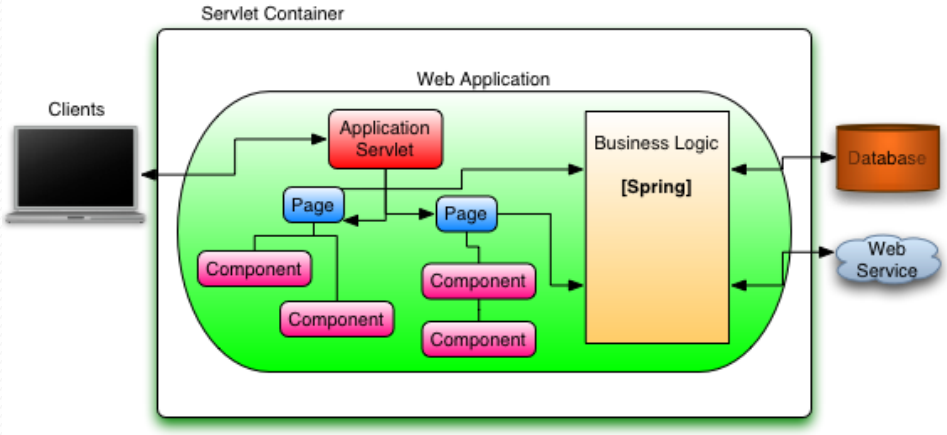
3. Frameworks

3.4 Tapestry

- Orientación a eventos y componentes

- ✓ Construcción de páginas web a través de plantillas
- ✓ Abstracción sobre URLs y parámetros
- ✓ Inversión de control, inyección de dependencias
- ✓ Trasformación de clases POJO en objetos complejos
- ✓ Varios lenguajes de programación
- ✓ Uso de tuberías o cadenas de filtros
- ✓ Convención sobre configuración
- ✓ Alta capacidad de reutilización

- x Escasa bibliografía



apache tapestry 5
Code less, deliver more.

4. El framework Myst

4.1 Análisis (I)

Decisiones arquitectónicas

- Orientado a acciones y URLs
  - ✓ Clara separación Vista – Modelo
  - ✓ Acciones y vistas controlables desde un módulo central: aplicación modificable sin cambios en la lógica
  - ✓ Componentes reutilizables
- Basado en configuración
  - ✓ Abundante bibliografía
  - ✓ Navegabilidad fácilmente describible por reglas
  - ✓ No hay que memorizar reglas especiales de programación
  - x Necesidad de archivo de configuración

Patrones utilizados

- MVC – arquitectura
- Service to Worker – solución global
- Context Object – comunicación de clases
- Singleton - despachador
- Command
- Intercepting Filter
- Composite View - HTML

4. El framework Myst

4.1 Análisis (y II)

Funcionalidades

- Control declarativo del flujo → Archivo de configuración
- Validación automática de la entrada
- Inversión de control → Motor de inyección de dependencias
- Etiquetas para controlar las vistas
- Extensibilidad

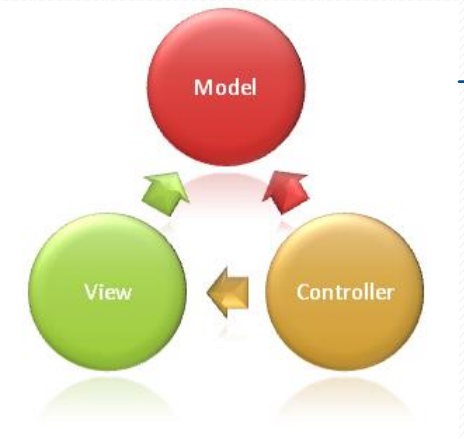
Filtros ←
- Internacionalización
- Procesamiento previo y posterior a la acción
- Gestión de errores

Aplicaciones poco acopladas ←
- Objetos POJO

4. El framework Myst

4.2 Diseño (I)

Arquitectura MVC



Controlador

- FrontController: Servlet, núcleo
- ApplicationController: despachador
- CommandMapper: relaciona peticiones con Actions
- Action: objetos POJO
- Filter: objetos POJO

Vista

- ViewHelper: encapsula las vistas, accesibles por Tags
- View: mapea el nombre de una vista con la vista real
- Tecnologías soportadas: JSP, HTML

Modelo

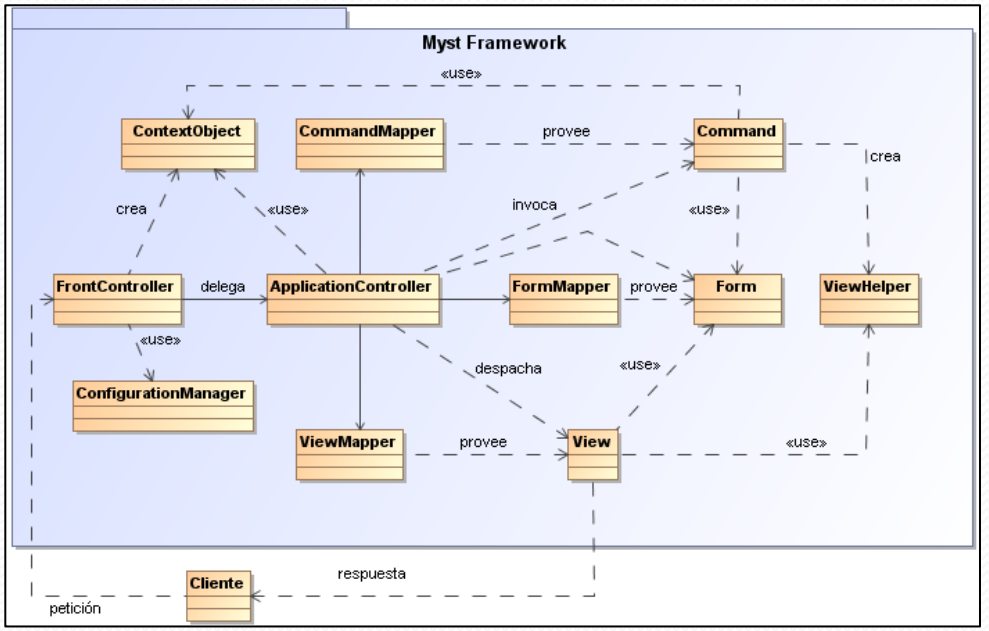
- No definido, cualquier tecnología

4. El framework Myst

4.2 Diseño (II)

Clases principales

- FrontController, ApplicationController, ContextObject, ViewHelper, Command, Filter: como los patrones
- ConfigurationManager: carga en memoria la configuración de la aplicación
- ExceptionHandler: gestión de las incidencias
- Form: formulario de entrada
- View: genera la vista a mostrar
- Mappers: contienen los comandos, form, vistas, etc. correspondientes a la petición
- I18N: provee internacionalización

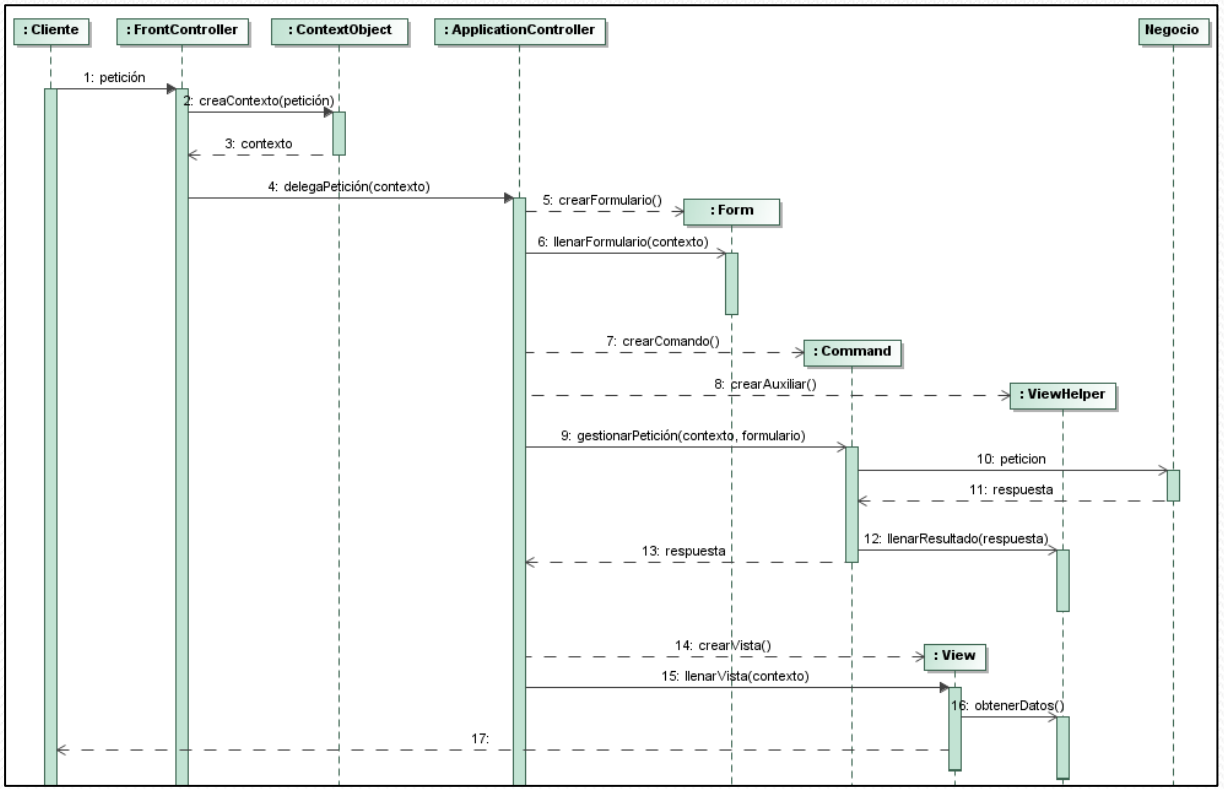


4. El framework Myst

4.2 Diseño (y III)

Diagrama simplificado de una petición

1. Llega una petición
2. Se crea el contexto
4. Se delega en el controlador de aplicaciones
5. Se carga el formulario
7. Se crea el comando
8. Se crea el auxiliar
9. Se ejecuta la petición, que accede a la lógica
12. Se carga el resultado en el auxiliar
14. Se crea la vista adecuada
15. La vista muestra los datos del auxiliar al cliente



4. El framework Myst

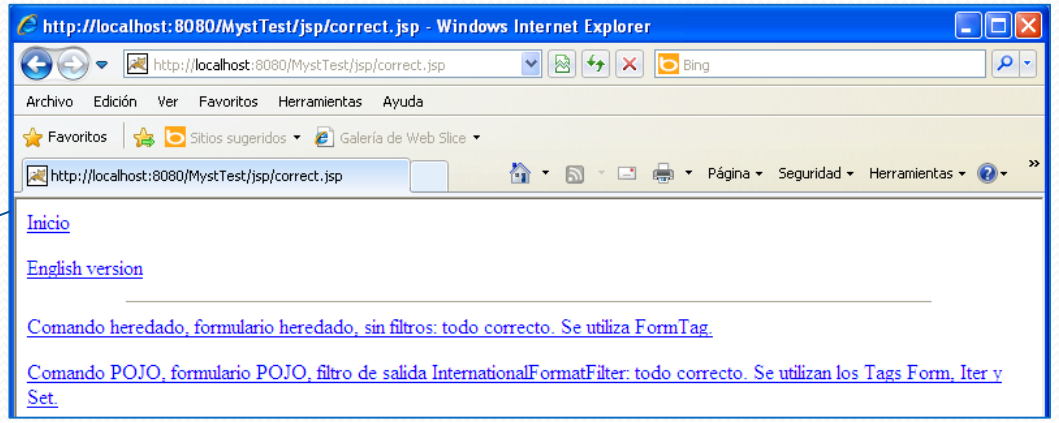
4.3 Aplicación para pruebas: MystTest

Motivos

- x Un framework no es una aplicación completa
- x Dificultad de testar el framework
- x El núcleo no es comprobable sin una aplicación
- x Las pruebas de integración de aplicaciones necesitan objetos virtuales

Resultados

- ✓ Todas las pruebas superadas
- ✓ Funcionamiento esperado
- ✓ Usa comandos y formularios POJO
- ✓ Correcta gestión de excepciones
- ✓ Correcta internacionalización



4. El framework Myst

4.4 Aplicación de muestra: HelpDesk (I)

Problemática

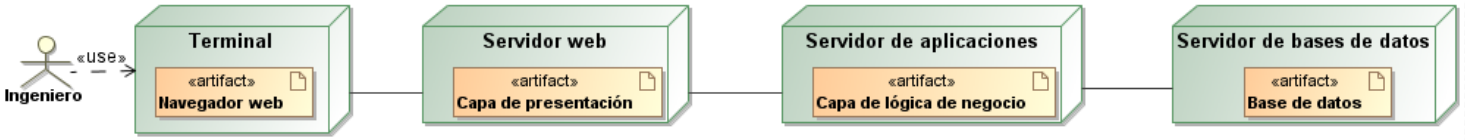
- Diseñar un programa para la gestión de incidencias informáticas

Requisitos

- Soporte para accesos simultáneos
- Dos tipos de usuarios: ingeniero y administrador
- Aplicación separada en servidores

Análisis

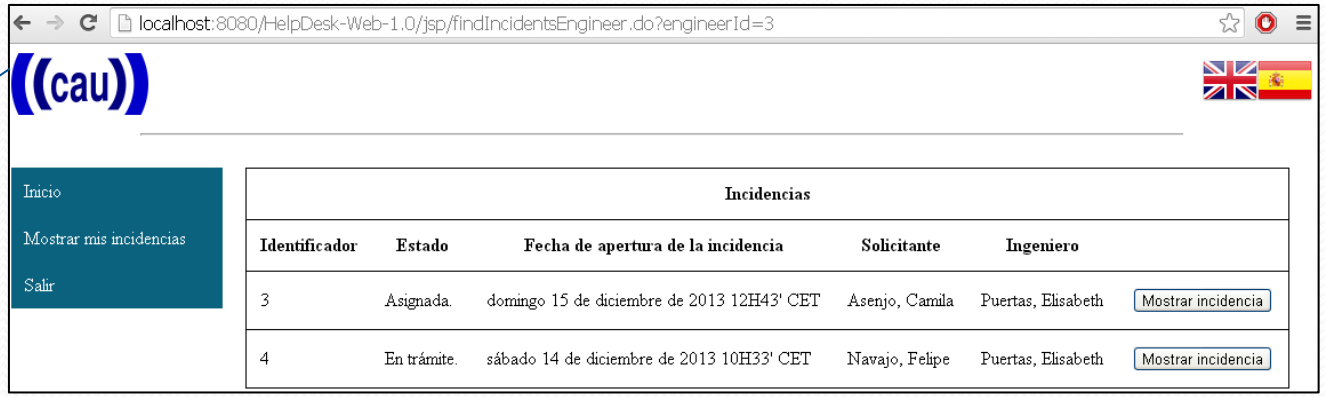
- Login para identificar usuarios
- Incidencias con estados
- Métodos para modificar incidencias
- Administrador usa métodos adicionales
- Restricciones por estado de incidencia y usuario
- Aplicación desplegada en tres capas



4. El framework Myst

4.4 Aplicación de muestra: HelpDesk (y II)

Diseño		Implementación
• Modelo-Vista-Controlador : framework Myst	Capa de presentación	• Comandos, formularios, vistas y excepciones en JBoss
• Enterprise JavaBean 3.1 + JNDI	Capa de lógica del negocio	• Bean de sesión con interfaz remota en JBoss
• Persistentes gracias a JPA 2.0: Hibernate	Capa de datos	• Base de datos relacional en MySQL



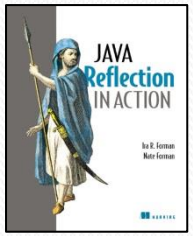
5. Conclusiones

Frameworks

- ¿Cuál es su futuro? ¿Cuál la meta?
- Continúan en plena evolución

Aprendizaje personal

- Frameworks
- Java: reflexión
- EJB 3.1



Dificultades

- Frameworks: información poco estructurada
- Reflexión
- Tags
- Aplicaciones: plugin Maven

Mejoras futuras

- Convención sobre configuración
- Soporte AJAX
- Catálogo de filtros
- Pruebas más completas
- ¿Orientación a eventos?