Diseño e implementación de un framework de presentación

Para aplicaciones Web 'Thin Client' en Java EE

PFC – Ingeniería Informática 2º Ciclo
Alumno: **Alejandro Marmelo Insua**
Consultor: **Óscar Escudero Sánchez**

UOC Universitat Oberta de Catalunya | 20 años

- Introducción
- Plataforma Java EE
- Frameworks de presentación
- Framework JavaMVC
- Aplicación Phonebook
- Conclusiones

Introducción

- Objetivos
- Planificación

Introducción

Objetivos

Profundizar en el conocimiento de la **plataforma Java EE**, en lo que a aplicaciones Web se refiere**,** a través del estudio de**:**

- **Los componentes Web** disponibles para su construcción.
- **Las arquitecturas de implementación**.
- **El catálogo de patrones de diseño recomendados** en la capa de presentación.

Estudio de los **principales frameworks de presentación** del mercado centrándose especialmente en sus funcionalidades y arquitectura.

Diseño e implementación de un **framework de presentación para aplicaciones Web Thin Client en Java EE** que proporcione las funcionalidades básicas de este tipo de herramientas.

Construcción de una aplicación que muestre el funcionamiento del framework.

Introducción

Planificación

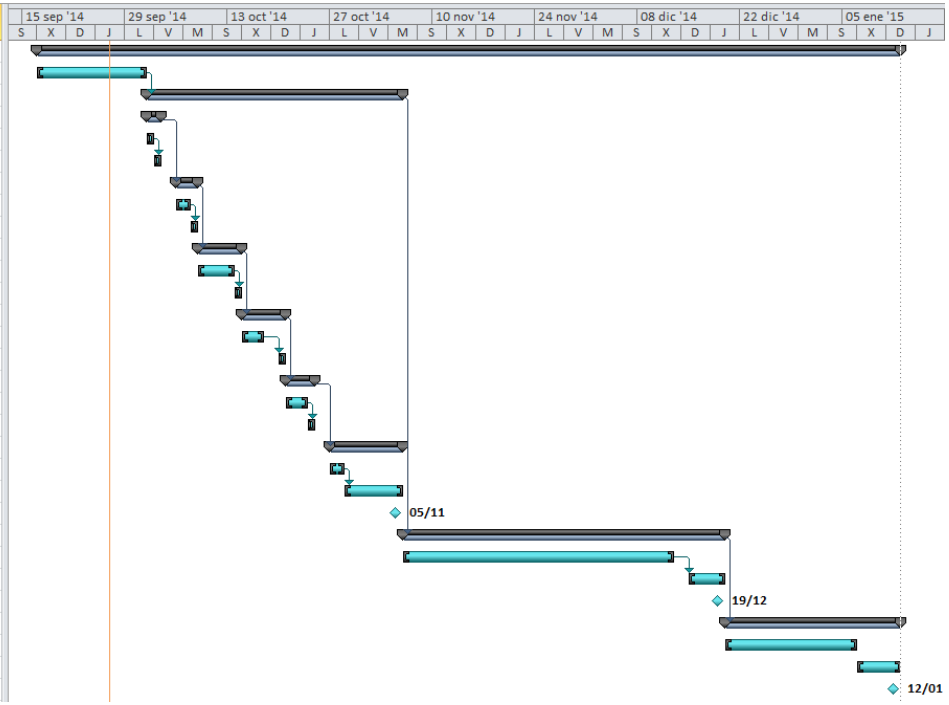
Plan de trabajo
17 sept 2014
1 oct 2014

Análisis y diseño
2 oct 2014 –
5 nov 2014

Implementación
6 nov 2014 –
19 dic 2014

Entrega final
20 dic 2014 –
12 ene 2015

Nombre de tarea	Duración	Comienzo	Fin
Proyecto de fin de carrera	**84 días**	**mié 17/09/14**	**lun 12/01/15**
Plan de trabajo - PEC1	11 días	**mié 17/09/14**	**mié 01/10/14**
Análisis y diseño - PEC2	**25 días**	**jue 02/10/14**	**mié 05/11/14**
Tecnología Web JEE	**2 días**	**jue 02/10/14**	**vie 03/10/14**
Investigación	1 día	jue 02/10/14	jue 02/10/14
Documentación	1 día	vie 03/10/14	vie 03/10/14
Framework Struts	**3 días**	**lun 06/10/14**	mié 08/10/14
Investigación	2 días	lun 06/10/14	mar 07/10/14
Documentación	1 día	mié 08/10/14	mié 08/10/14
Framework Struts 2	**4 días**	**jue 09/10/14**	**mar 14/10/14**
Investigación	3 días	**jue 09/10/14**	lun 13/10/14
Documentacion	1 día	mar 14/10/14	mar 14/10/14
Framework Spring Web MVC	**4 días**	**mié 15/10/14**	lun 20/10/14
Investigación	3 días	mié 15/10/14	vie 17/10/14
Documentación	1 día	lun 20/10/14	lun 20/10/14
Framework JSF	**4 días**	**mar 21/10/14**	vie 24/10/14
Investigación	3 días	mar 21/10/14	jue 23/10/14
Documentación	1 día	vie 24/10/14	vie 24/10/14
A/D Framework presentación	**8 días**	**lun 27/10/14**	**mié 05/11/14**
Análisis	2 días	lun 27/10/14	mar 28/10/14
Diseño	6 días	mié 29/10/14	mié 05/11/14
Entrega análisis y diseño	0 días	**mié 05/11/14**	**mié 05/11/14**
Implementación - PEC3	**32 días**	**jue 06/11/14**	**vie 19/12/14**
Implementación framework	27 días	jue 06/11/14	vie 12/12/14
Desarrollo aplicación de prueba	5 días	lun 15/12/14	vie 19/12/14
Entrega implementación	0 días	**vie 19/12/14**	**vie 19/12/14**
Entrega final	**17 días**	**sáb 20/12/14**	**lun 12/01/15**
Memoria	13 días	**sáb 20/12/14**	mar 06/01/15
Presentación	4 días	mié 07/01/15	lun 12/01/15
Proyecto de fin de carrera entregado	**0 días**	lun 12/01/15	lun 12/01/15



Plataforma Java EE

- Introducción
- Arquitecturas de implementación de aplicaciones Web
  - ✓ Arquitectura JSP-Modelo 1
  - ✓ Arquitectura JSP-Modelo 2
- Patrones de diseño de la capa de presentación

Plataforma Java EE

Introducción

- **Java Enterprise Edition** es una plataforma de programación utilizada para desarrollar y ejecutar aplicaciones Java.

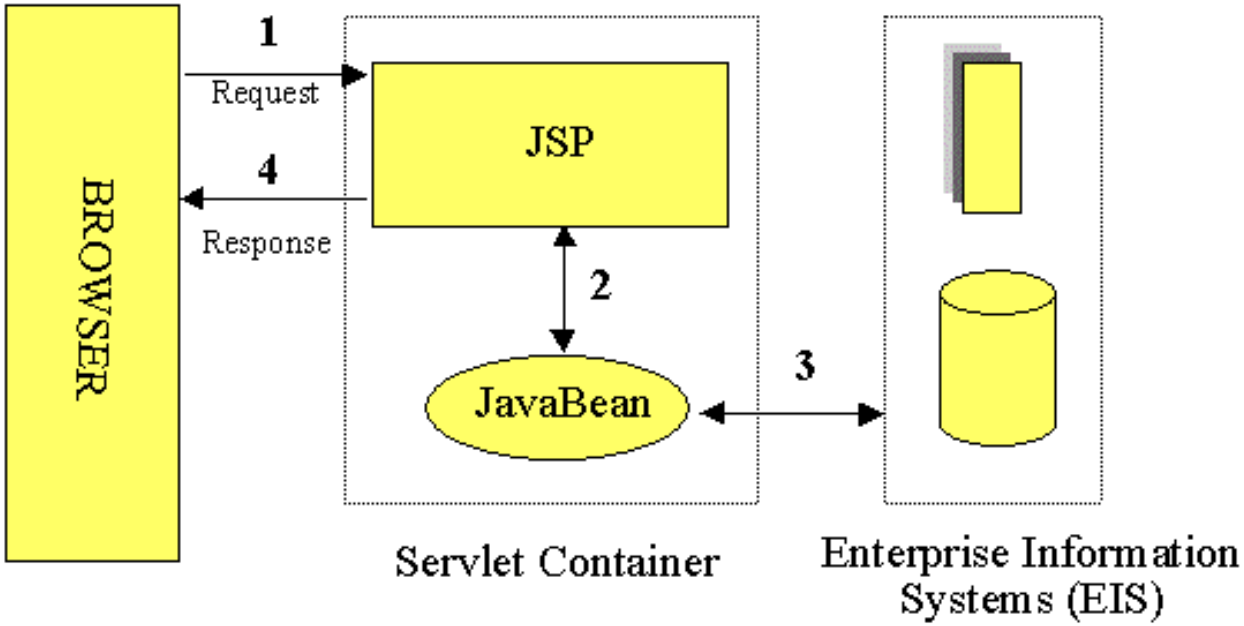
- Las aplicaciones Java EE se construyen en base a **Componentes** (Clientes de aplicación y Applets, **Componentes Web** (Servlet, JSP y JSF) y Componentes de negocio (EJB)).

- Los **Componentes** se ensamblan en **Módulos** Java EE **(JAR, WAR o EAR)** y se ejecutan en **Contenedores** que pueden ubicarse en la máquina cliente (Applets y Clientes de aplicación) o en un servidor Java EE (**Componentes Web** y de Negocio).

- Una **aplicación Web** es una extensión dinámica de un servidor Web o aplicación llevada a cabo mediante **Componentes Web**.

Arquitecturas de implementación de aplicaciones Web

Arquitectura JSP Modelo-1:



Arquitecturas de implementación de aplicaciones Web

Arquitectura JSP Modelo-1:

Caracterizada por dar a la JSP la responsabilidad tanto de procesar la petición entrante como de generar la respuesta del cliente.

Ventajas

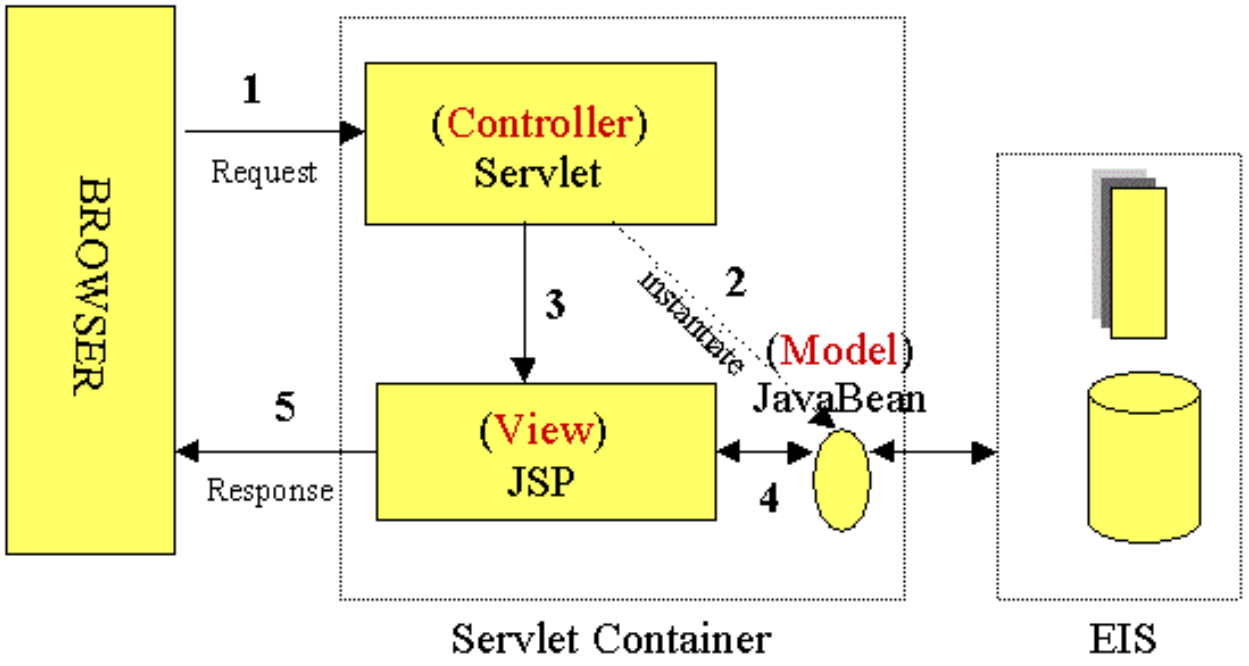
- Resulta más rápida de implementar que arquitecturas más elaboradas

Inconvenientes

- Conduce al uso indiscriminado de scriptlets en las JSP.
- Complica tanto la creación como el mantenimiento de las JSP.
- Dificulta la participación de diseñadores en el desarrollo.
- Cada JSP es responsable de gestionar el estado de la aplicación, seguridad...
- Aplicable únicamente en aplicaciones muy sencillas

Arquitecturas de implementación de aplicaciones Web

Arquitectura JSP Modelo-2:



Arquitecturas de implementación de aplicaciones Web

Arquitectura JSP Modelo-2:

Enfoque híbrido que aprovecha los puntos fuertes de la tecnología **Servlet** y **JSP**.

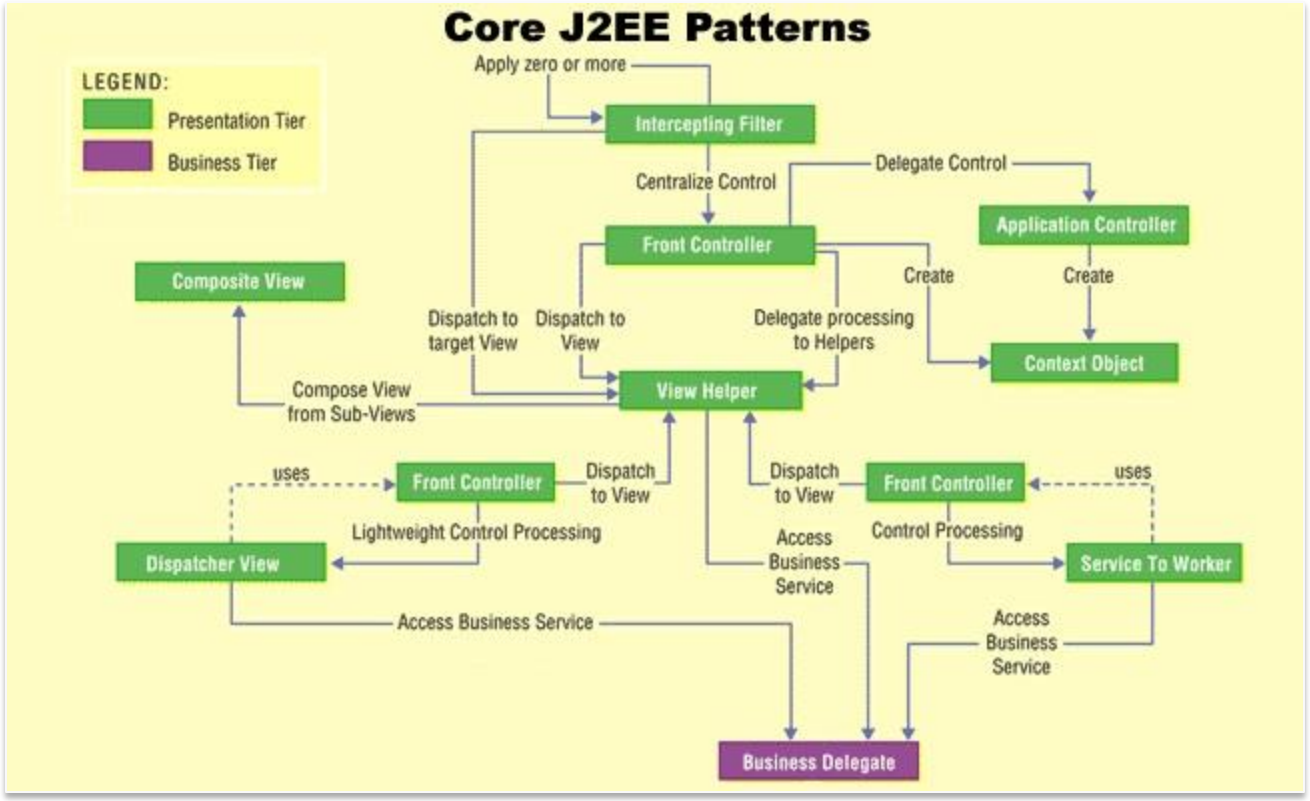
- Las JSP se utilizan para generar las vistas.
- Los Servlet para las tareas de procesado intensivo.

Conocida como **MVC2** o **Web MVC** dónde:

- El Servlet actúa como **Controlador** - Puente entre la Vista y el Modelo.
- La JSP actúa como **Vista** - Responsable de presentar los datos.
- Los JavaBean actúan como el **Modelo** - Lógica y procesos de negocio.

Ventajas	Inconvenientes
• Facilita la modificación del Modelo, Vista y Controlador por separado. • Permite la separación de tareas en los equipos de desarrollo.	• La división del sistema puede suponer un sobrecosto en la comunicación entre el modelo y la vista.

Patrones de diseño de la capa de presentación



Patrones de diseño de la capa de presentación

- **Intercepting Filter:** Permite decorar el tratamiento de las peticiones recibidas con un pre-procesamiento y un post-procesamiento adicional.
  - Autenticación de usuarios.
  - Cifrado/Descifrado de la petición.
  - Compresión/Descompresión de la petición

- **Front Controller:** Proporciona un único punto de entrada a la capa de presentación.
  - Manejo de la seguridad.
  - Validación.
  - Control de flujo.

- **Application Controller:** Extrae del componente Front Controller la gestión de invocación a las acciones y la gestión de envío a las vistas.

Patrones de diseño de la capa de presentación

- **Context Object:** Permite pasar datos de objetos de un contexto específico a otro, sin pasar esos objetos fuera de su contexto.
- **View Helper:** Minimiza la cantidad de código Java en las JSP simplificando el desarrollo y mejorando la mantenibilidad.
- **Composite View:** Proporciona la forma de crear vistas combinando fragmentos de otras vistas.
- **Service To Worker:** Combinación de los patrones View Helper, Front Controller y Application Controller.

Frameworks de presentación

- Struts
- Struts^2
- JavaServer Faces
- Spring

Frameworks de presentación

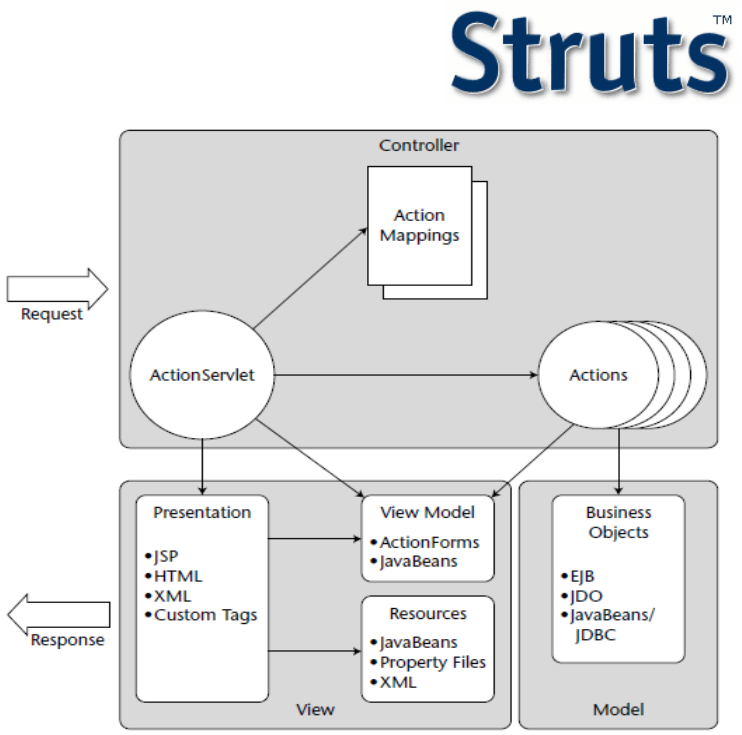
Struts

Ventajas

- Orientado a acciones y URLs.
- Proporciona su propio componente controlador.
- Utiliza otras tecnologías para la Vista y el Modelo.
- Validación de datos mediante ActionForm o DynaBeans.
- Librería de etiquetas.
- Amplia bibliografía.

Inconvenientes

- Configuración a través de fichero XML.
- Actions y ActionForms no son POJO.
- Action implementa el patrón singleton.
- Proyecto abandonado.

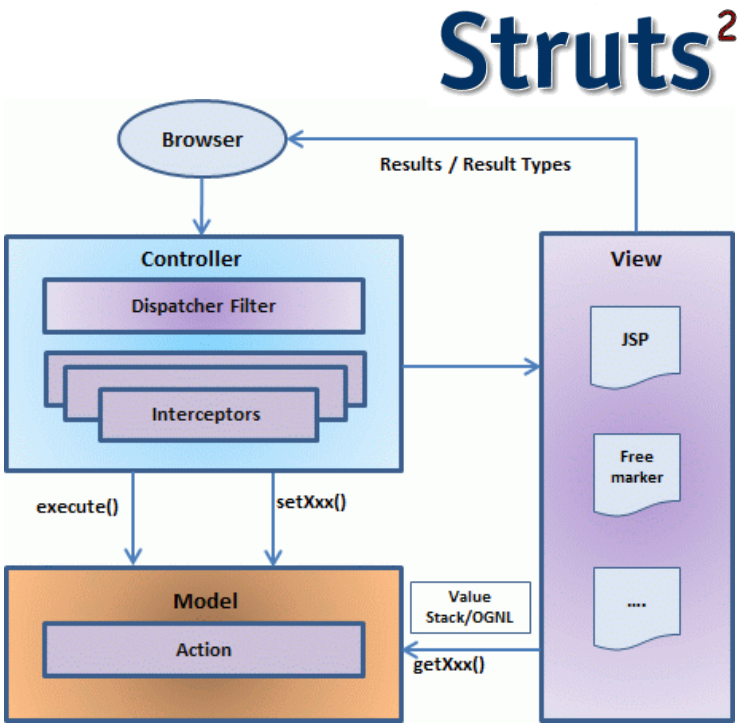


Frameworks de presentación

Struts^2

Ventajas

- Orientado a acciones y URLs.
- Los Action no son singleton.
- Cualquier POJO con el método execute puede ser un Action.
- Facilidad de testeo. Se elimina o reduce la necesidad de acceder a objetos request y response.
- Permite el uso de filtros e interceptores.
- Utiliza OGNL para las conversiones de tipos.
- Tags con soporte Ajax.
- Configurable vía XML y con anotaciones.
- Acceso a información del contexto a través de Value Stack.



Inconvenientes

- Difícil migración de Struts a Struts^2.
- Documentación limitada.

Frameworks de presentación

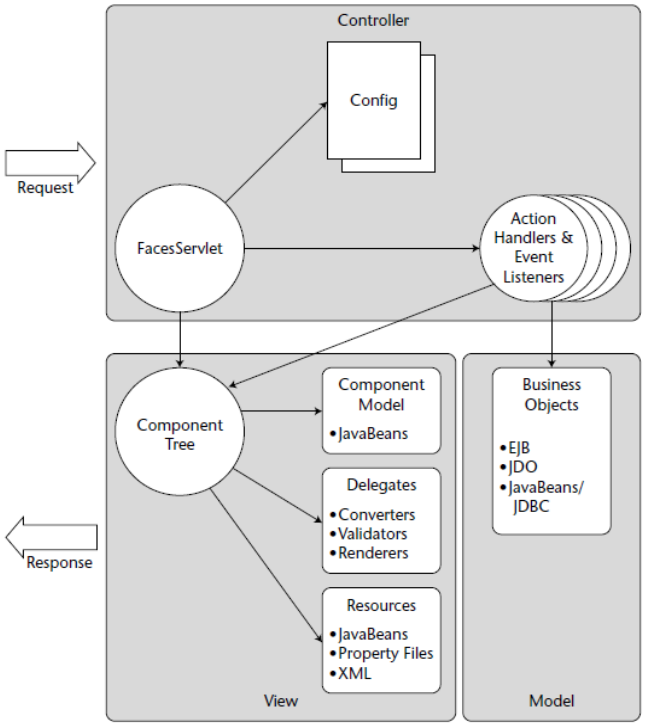
JavaServer Faces

Ventajas

- Basado en componentes.
- Proporciona un entorno MVC más rico y cercano al MVC tradicional.
- Soporte nativo de Ajax.
- Forma parte de la especificación Java EE desde la versión 1.4.

Inconvenientes

- Curva de aprendizaje alta dado que suele ser necesario utilizar librerías de componentes externas si se quieren llevar aplicaciones con cierta complejidad.
- Páginas JSP plagadas de etiquetas.
- Su naturaleza como estándar hace que no pueda evolucionar tan rápido como en otros entornos.



Frameworks de presentación

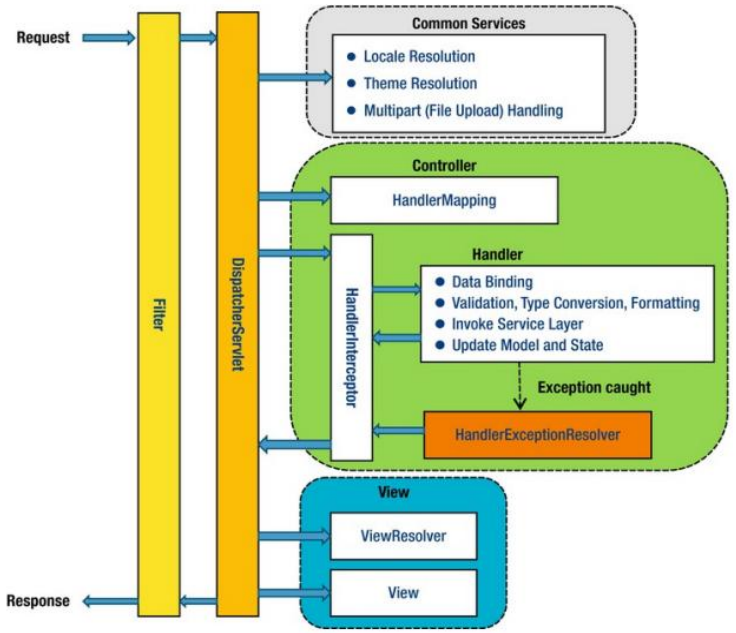
Spring MVC

Ventajas

- Orientado a acciones y Urls.
- Utilización de objetos POJO.
- Potente motor de IoC e inyección de dependencias.
- Facilita la realización de pruebas.
- Amplia bibliografía.

Inconvenientes

- Sin soporte Ajax nativo.
- Curva de aprendizaje más pronunciada debido a la necesidad de conocer cómo funciona Spring (IoC, DI).
- Proporciona tantos tipos de controladores que puede resultar confuso saber cual debe utilizarse en cada caso.



Framework JavaMVC

- Requisitos
- Diseño
- Componentes del framework
- Producto resultante

Framework JavaMVC

Requisitos

- Simplificar y agilizar el desarrollo de la capa de presentación en aplicaciones Web Java EE siguiendo el patrón arquitectónico MVC.
- Configurar de forma declarativa los componentes relacionados con una petición.
- Proporcionar mecanismos para la internacionalización.
- Permitir el tratamiento personalizado de excepciones.
- Permitir interceptores a nivel de acción.
- Etiquetas que faciliten la separación entre la lógica de preparación y el formateo de los datos fuera de las páginas JSP.

Framework JavaMVC

Diseño

Patrones arquitectónicos:

- **Model-View-Controller** (en su versión adaptada para aplicaciones Web).

JavaMVC propone un desarrollo basado en el patrón Web MVC dónde el **Controlador** despacha las peticiones a componentes Action y se apoya en otras tecnologías como JSP y JavaBeans para la implementación del **Modelo** y la **Vista**.

- No impone restricciones sobre cómo implementar el **Modelo** de la aplicación permitiendo el uso de cualquier tecnología (EJB, JavaBeans, JDO, etc.).

- La **Vista** consiste en JSPs y un conjunto de etiquetas a medida que ayudan a eliminar la lógica que no está directamente relacionada con la presentación.

- El **Controlador** se configura a partir de la lectura de un XML en el que se declaran los elementos que participan en el procesamiento de cada petición.

Framework JavaMVC

Diseño

Patrones de diseño:

- **Service To Worker** (Front Controller, Application Controller y View Helper)

Centraliza el control y el manejo de las peticiones, para recuperar el modelo de presentación antes de dar el control a la vista. De la que se eliminará la lógica de preparación y formateo de los datos.

- **Context Object**

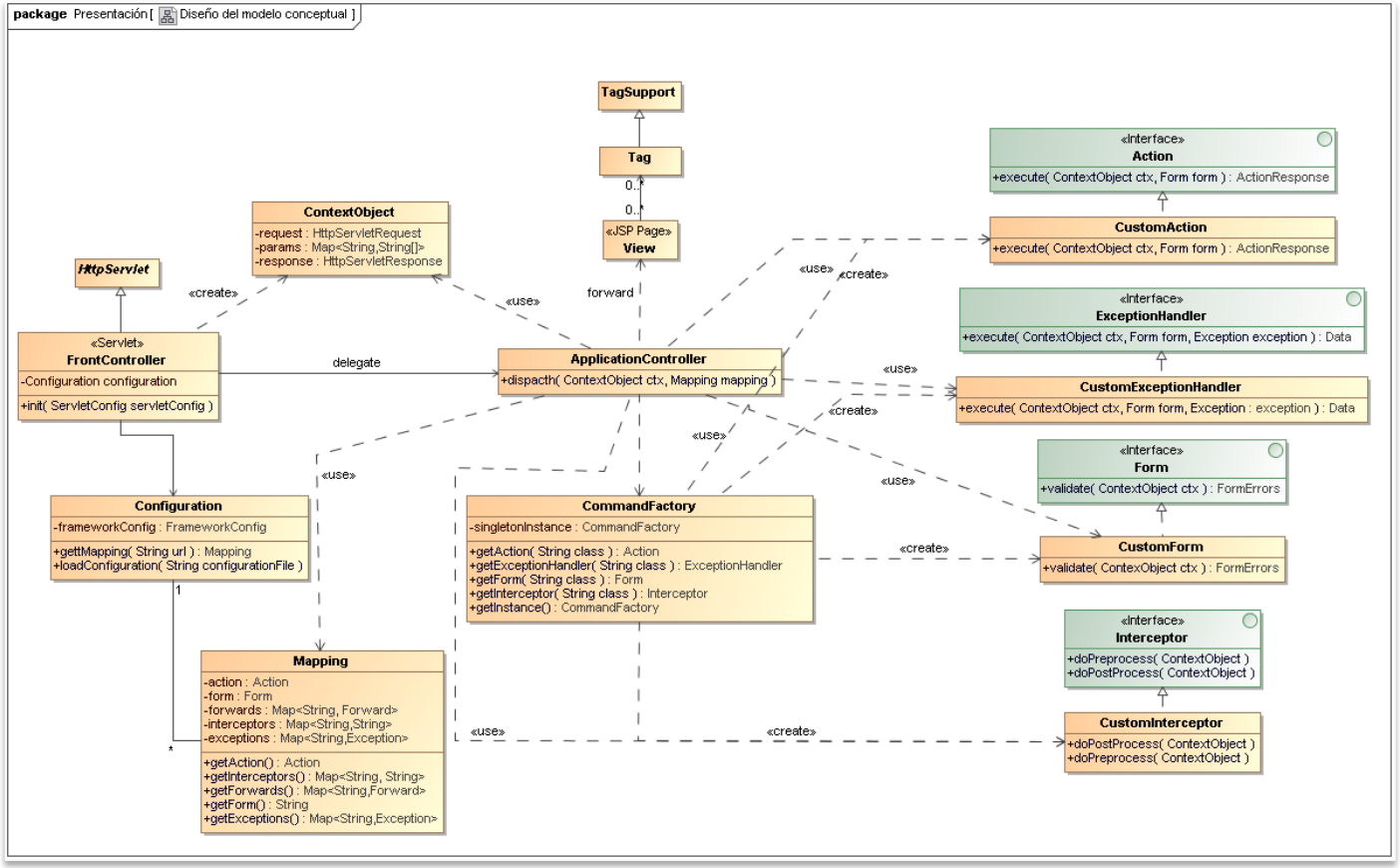
Utilizado para encapsular los datos específicos del contexto de cada petición y en la captura y validación de datos recibidos a nivel de formulario.

- **Singleton y Factoría** (concreta)

Utilizados para proporcionar al framework una única factoría encargada de instanciar los elementos declarados por el desarrollador en el documento XML de configuración .

Framework JavaMVC

Diseño



Framework JavaMVC

Componentes del framework

- **Formularios**. JavaBeans vinculados a una o varias **Acciones** que recogen en sus propiedades los datos recibidos con la petición y permiten su validación antes de que la **Acción** solicitada en la petición sea procesada.

- **Acciones**. Clases encargadas de redirigir el procesamiento a la lógica de negocio pertinente para finalmente entregar el resultado obtenido al recurso apropiado.

- **Interceptores**. Clases vinculadas a una o varias **Acciones** cuyo procesamiento se lleva a cabo antes y después de ejecutar la **Acción** solicitada en la petición.

- **Manejadores de excepciones**. Clases que permiten definir el comportamiento que tendrá la aplicación ante una situación de excepción.

- **Recursos para la internacionalización**. Paquete que contiene los ficheros de recursos que la aplicación utilizará para su internacionalización.

Framework JavaMVC

Componentes del framework

- **Librerías de etiquetas.** JavaMVC proporciona librerías de etiquetas que ayudan a eliminar de la Vista aquella lógica no directamente relacionada con la presentación.

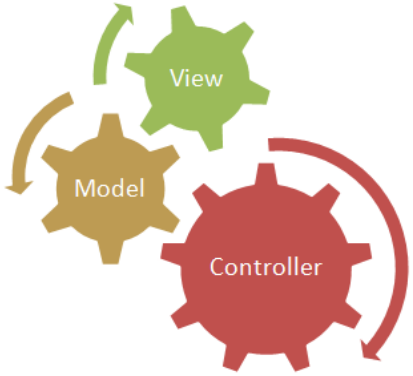
  Estas librerías proporcionan etiquetas que ayudan a:

  - **Internacionalizar aplicaciones**
    - ✓ **Error**. Etiqueta que permite mostrar y formatear un conjunto de mensajes de error como resultado a la validación de un formulario.
    - ✓ **Message**. Etiqueta destinada a la recuperación de mensajes internacionalizados.
    - ✓ **I18n**. Etiqueta destinada a la internacionalización de números, monedas, fechas y horas.
  - **Generar HTML**
    - ✓ **Form**. Etiqueta que permite la construcción de formularios HTML capaces invocar acciones declaradas en la configuración del framework.

Framework JavaMVC

Producto resultante

- Framework de presentación basado en peticiones **HTTP.**
- Configuración declarativa vía **XML**.
- **Internacionalizable**.
- **Librería de etiquetas** para la vista.
- Clara separación entre **Modelo, Vista** y **Controlador**.
- Permite el tratamiento de **excepciones** y el uso de **Interceptores**.
- Dependiente de las especificaciones **Servlet 3.0** y **JSP 2.1**



Taglibs

Aplicación Phonebook

- Análisis
- Diseño e Implementación
- Interfaz

Aplicación Phonebook

Análisis

Necesidad

- Agenda telefónica on-line que permita a cada usuario gestionar sus contactos.

Requisitos

- Un usuario externo a la aplicación podrá crear una cuenta en el sistema.
- Los usuarios del sistema deben proporcionar sus credenciales de acceso para acceder a su agenda telefónica.
- El sistema permitirá la gestión de los contactos (alta, baja, modificación y consulta).
- El sistema permitirá al usuario cambiar el idioma de la aplicación (Inglés < > Castellano).
- El sistema permitirá al usuario cambiar su clave de acceso al sistema.
- El sistema persistirá los datos de contactos y usuarios de forma no volátil.

Aplicación Phonebook

Diseño e Implementación

Estrategia

- **Cada caso de uso extraído de los requisitos corresponde a una acción diferente.**

  Phonebook utilizará componentes Action con los que servirá cada petición recibida por la aplicación.

- **Ciertas peticiones requieren la validación de los datos enviados antes de ejecutar la acción solicitada.**

  Phonebook utilizará componentes Form para capturar y validar los datos recibidos.

- **El resultado de toda petición debe devolver una respuesta al usuario.**

  Phonebook utilizará páginas JSP y etiquetas para mostrar el resultado obtenido en cada petición.

- **Las acciones permitidas siguen un flujo que define la navegación en la aplicación.**

  Phonebook configura la navegabilidad mediante un XML que relaciona componentes Action, Form y JSP. Este XML permite declarar los recursos que se utilizarán para internacionalizar la aplicación.

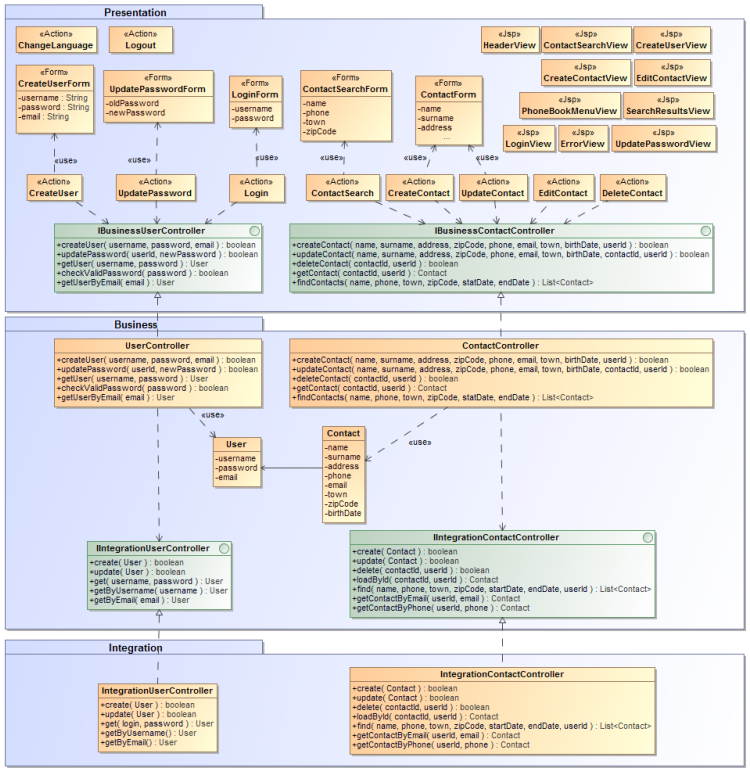
- **La naturaleza de la aplicación requiere la persistencia de la información que gestiona.**

  Phonebook utilizará el framework Hibernate para simplificar la persistencia.

Aplicación Phonebook

Diseño e Implementación

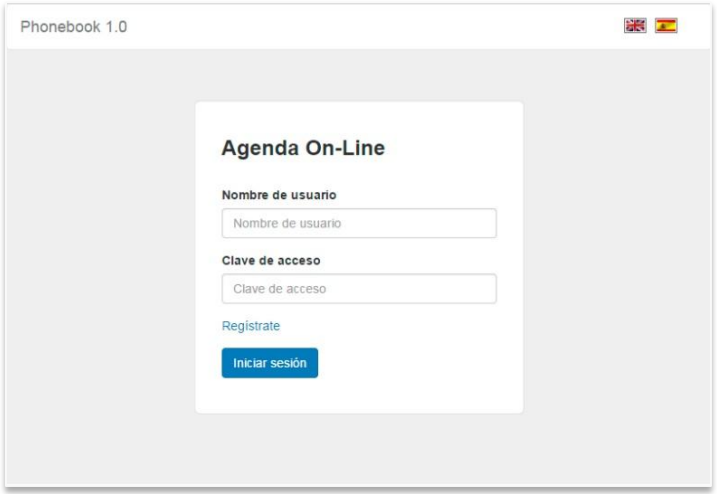
Arquitectura del sistema dividida en tres capas:

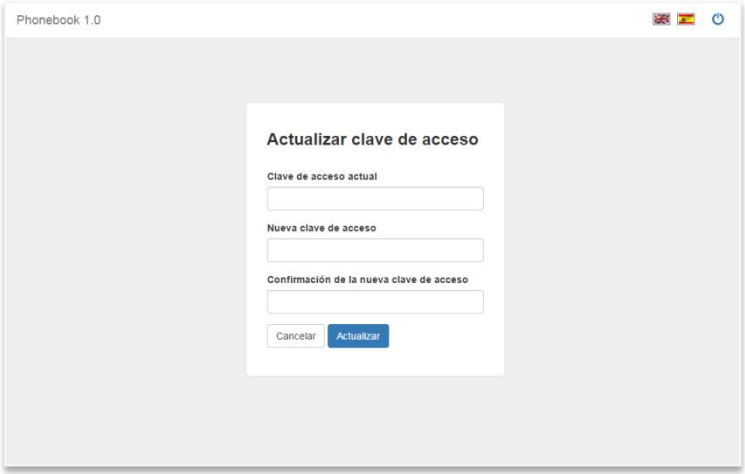


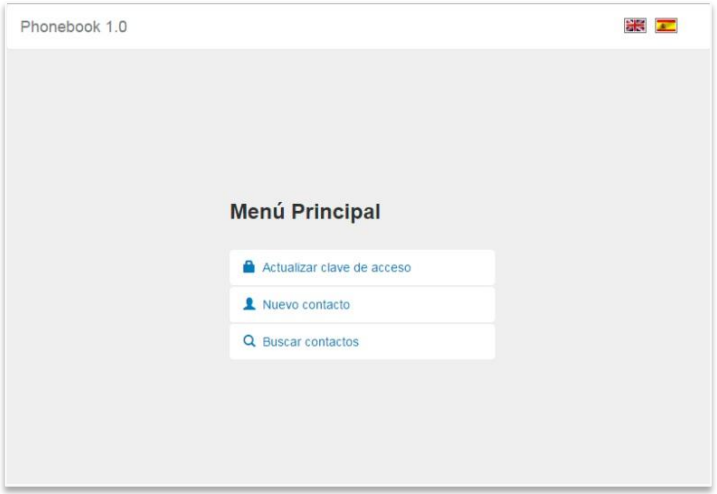
- **Presentación**. Aplica el patrón arquitectónico **MVC** en su variante adaptada a la arquitectura en capas.
  - ✓ Páginas JSP y etiquetas para la **Vista**.
  - ✓ Componentes Action y Form de soporte al **Controlador**.
  - ✓ Descriptor XML para declarar las relaciones entre componentes.
- **Negocio**. Formada por:
  - ✓ Controladores que implementan las operaciones necesarias por la capa de presentación.
  - ✓ JavaBeans que representan los conceptos detectados durante el análisis. **Modelo**.
- **Integración**. Formada por:
  - ✓ Controladores que implementan las operaciones necesarias por la capa de negocio.
  - ✓ Descriptor XML para el mapeo entre entidades y el modelo de objetos.

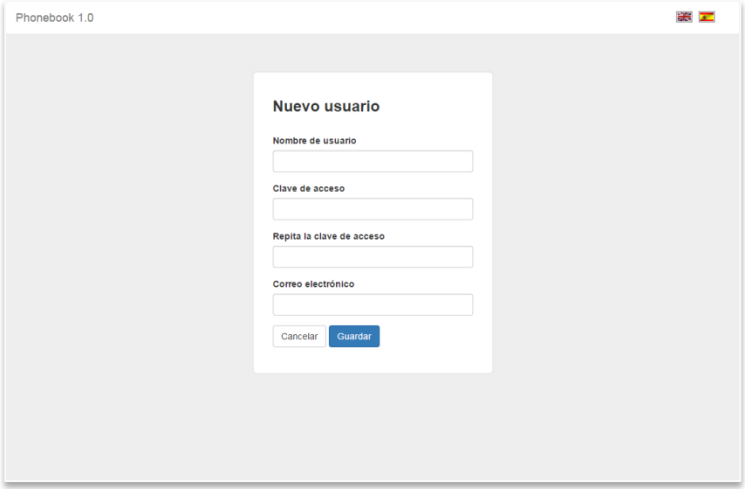
Aplicación Phonebook

Interfaz



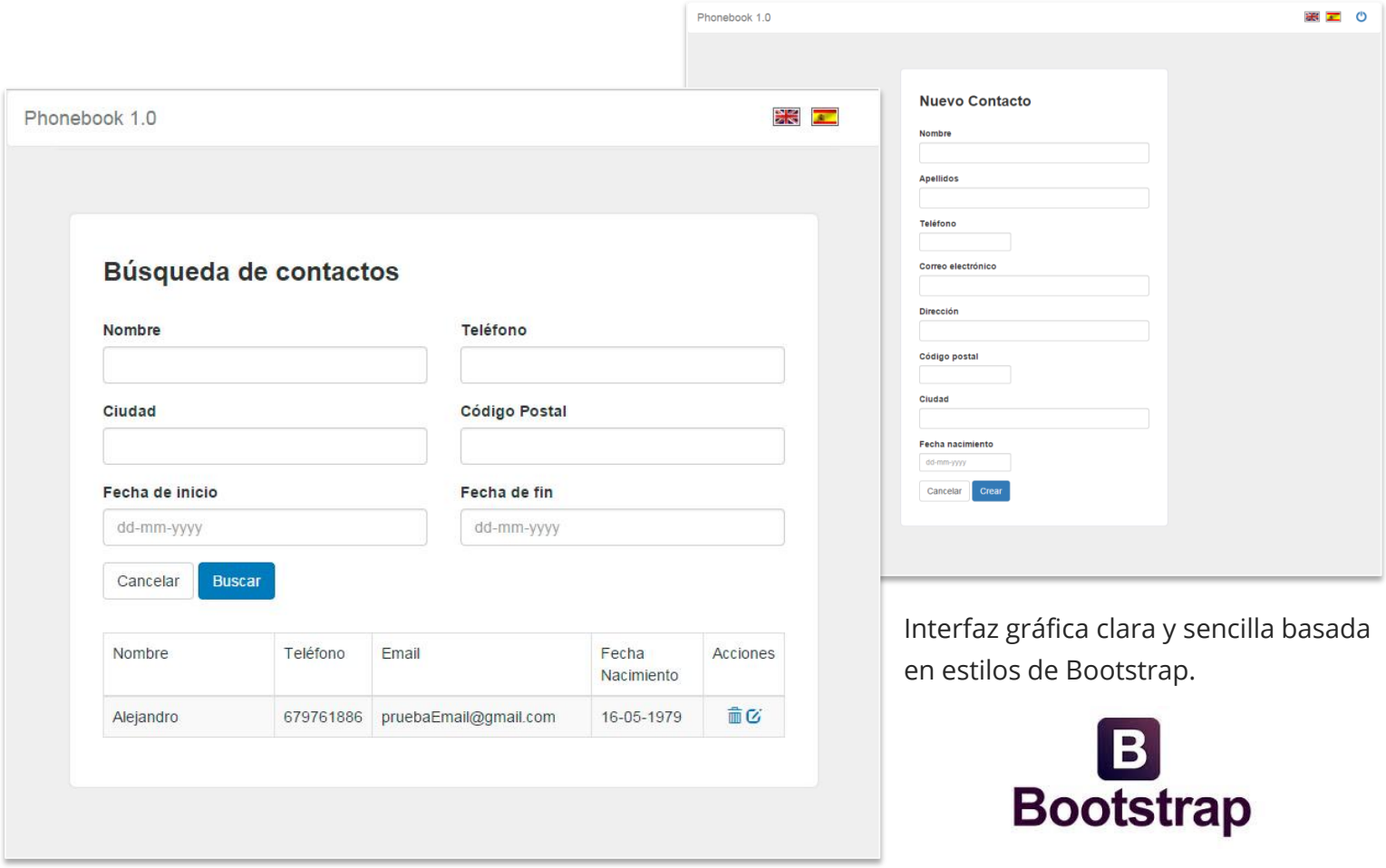






Aplicación Phonebook

Interfaz



Interfaz gráfica clara y sencilla basada en estilos de Bootstrap.

Conclusiones

Conclusiones

Conclusiones

Sobre los frameworks

- Soluciones ampliamente adoptadas que no frenan la aparición de nuevas alternativas.
- Todo framework tiene sus ventajas e inconvenientes, no existe una solución perfecta.
- Continua evolución.

Sobre el proyecto

Aprendizaje

- Frameworks de gran aceptación en el mercado.
- JAXB, Reflection, Tags, Internacionalización.

Dificultades

- Documentación escasa, mal estructura o innecesariamente compleja.
- Aprendizaje de nuevas tecnologías para implementar el framework.
- Timming ajustado para el alcance del proyecto.

Mejoras futuras

- Soporte para el envío de ficheros.
- Uso de anotaciones.
- Creación de formularios dinámicos de forma declarativa.
- Librería de etiquetas con mayor funcionalidad (incluyendo soporte Ajax).
- Evolucionar el framework buscando una mayor abstracción de la tecnología Servlet.
- Etc.

Gracias

por su atención

PFC – Ingeniería Informática 2º Ciclo
Alumno: **Alejandro Marmelo Insua**
Consultor: **Óscar Escudero Sánchez**

UOC Universitat Oberta de Catalunya | 20 años