PROYECTO FIN
DE CARRERA

Creación de un *framework* de presentación para aplicaciones JEE

Autor: Alberto Díaz Martínez
Consultor: Oscar Escudero Sánchez

Agenda

1.ESTUDIO JEE.
    1.1.El estándar JEE.
    1.2.Patrones y frameworks.
    1.3.Frameworks JEE.

2.FORMWORK. UN FRAMEWORK DE CAPA DE PRESENTACIÓN.
    2.1.Análisis y diseño.
    1.Implementación.

3. CONCLUSIONES

I JEE. El estándar JEE

- JEE (*Java Enterprise Edition*) es una colección de especificaciones que publica el JCP (*Java Community Process*) para la creación de aplicaciones empresariales sobre la plataforma Java.
- Actualmente engloba 32 especificaciones diferentes que tratan aspectos como transacciones, *servlets*, EJB, proceso de XML, presentación, persistencia, etc.
- Las aplicaciones JEE deben correr sobre un servidor de aplicaciones que cumpla con la especificación. En el mercado existen varios, tanto libres (JBoss, Geronimo, Glassfish) como propierarios (IBM WAS, Oracle WebLogic)
- La última especificación publicada es JEE 6 (JSR 316). La implementación de referencia es el servidor Glassfish 3.x. El estándar para la capa de presentación en JEE 6 es JSF 2.0 (JSR 314)

I JEE. Patrones y frameworks

- Un patrón es una receta probada repetidamente con éxito en la solución de un problema recurrente en el ámbito de una actividad, en este caso, en el desarrollo de software.
- El patrón describe un problema y su solución, con una serie de diagramas UML que se deberán implementar para cada caso concreto. MVC y Front Controller son ejemplos de patrones ampliamente usados en JEE.
- Un framework es una herramienta formada por unas bibliotecas de clases y unas reglas de buenas prácticas que facilitan la aplicación de patrones, permitiendo al desarrollador abstraerse de los aspectos tecnológicos para centrarse en el negocio.

I. JEE. Frameworks JEE

- Dividimos el estudio de *frameworks* de capa de presentación en clásicos y AJAX según hagan uso o no de esta tecnología.

- CLÁSICOS
  - JSP + Servlets
  - Struts 1 y 2
  - JSF 1.x
- AJAX
  - JSF 2.x
  - ZK
  - ItsNat.

I JEE. Servlets + JSP

- Si bien estrictamente hablando no se trata de un *framework,* el uso combinado de *servlets* y páginas JSP (modelo 2 de arquitectura) supone el primer éxito aplicando el patrón MVC.
- Las páginas JSP actúan como vistas (V), mostrando al usuario los datos (una colección de POJOS) y trasladando los gestos de éste hacia el controlador adecuado.
- Los *servlets* interpretan el papel de controlador (C). Reciben los gestos del usuario en forma de peticiones HTTP, actúan sobre el modelo, preparan los datos de respuesta y seleccionan la siguiente vista.
- El modelo es de implementación libre. Todo lo relacionado con peticiones, respuestas y protocolos es ajeno al modelo, que solamente debe mantener los datos de la aplicación.

I JEE. Struts 1

- Struts 1 fue el primer *framework* de éxito. Llegó a alcanzar el estatus de estándar *de facto*.
- Creado por Steve McClanahan, quien lo donó a la fundación Apache.
- Las vistas se implementan con JSP. Incorpora una serie de librerías de etiquetas para ayudar en su construcción
- Usa un *servlet* como *Front Controller*. Los controladores se crean como subclases de la clase `Action`. Ejecutan la acción sobre el modelo y seleccionan la siguiente vista.
- La navegación entre páginas se define en el fichero de configuración `struts-config.xml`. Se mapean todas las posibles páginas a las que se puede llegar desde una acción en concreto.

I JEE. Struts 2

- Struts 2 es un *framework* nacido de la fusión de Struts 1 con WebWorks.
- No usa *servlets,* sino filtros, interceptores, acciones y resultados.
- Las vistas se implementan con JSP.
- Usa un filtro como *Front Controller*. Los controladores se crean como clases que implementa la interfaz `Action`. Estas clases son POJOS. Cualquier método que devuelva `String` puede ser un método de acción.
- La navegación entre páginas se define en el fichero de configuración `struts.xml`.
- La petición y la respuesta pasarán por una pila de interceptores antes de llegar a la clase de acción o a la página JSP respectivamente.
- Estos interceptores realizan tareas como validar, extraer parámetros etc.

I JEE. JSF 1.x

- JSF es el estándar JCP para la capa de presentación de aplicaciones JEE. Aplica patrón MVC.
- Introduce el concepto de componente reutilizable.
- Vistas con JSP y librerías de etiquetas. Controladores con POJOS que siguen la convención *Java Beans*.
- Las reglas de navegación se definen en el fichero de configuración `faces-config.xml`.
- Las peticiones siguen un ciclo de vida con seis fases:
  - Restore view
  - Apply request values
  - Process validations
  - Update model values
  - Invoke application
  - Render response

I JEE. JSF 2.x

- JSF es la evolución de JSF 1.
- Vistas con *facelets* (XHTML) y namespaces. Controladores con POJOS que siguen la convención *Java Beans*.
- Mediante y anotaciones y mapeo de *outcomes* a *facelets* se hace innecesario definir reglas de navegación en el fichero de configuración `faces-config.xml`.
- Introduce soporte AJAX mediante la etiqueta `<f:ajax />`.
- Las peticiones siguen el mismo ciclo de vida en seis fases. En las peticiones AJAX estas 6 fases se aplican solamente a los componentes involucrados.

I JEE. ZK

- ZK es un *framework* AJAX puro, pero sin tener que escribir una sola línea de código `JavaScript`.
- Basado en componentes y eventos. Aplica el paradigma SPI (*Single Page Interface*) y los patrones MVC y MVVM.
- Vistas con ZUML, un lenguaje de marcas basado en XML.
- Controladores mediante clases que implementan `SelectorComposer` (si se usa MVC) o POJOS (si se emplea MVVM).
- Server + Client fusion. Cada componente en el cliente (o *widget*) tiene su par en el servidor (o componente) y ambos actúan como uno solo.
- Ofrece tanto al desarrollador como al usuario una experiencia cercana a la de las aplicaciones de escritorio.

I JEE. ItsNat

- ItsNat es un *framework* AJAX que incorpora algunas ideas nuevas.
- Solo utiliza tecnologías web estándar del W3C: DOM, (X)HTML y CSS.
- Nada de lógica en el cliente. Todo se hace en el servidor mediante el API W3C DOM.
- Se simula un *browser* en el servidor con el mismo contenido DOM que en el cliente: aproximación TBITS (*The Browser Is The Server)*
- Se aprovecha gran parte del API de Swing para los componentes y eventos.
- Debemos extender un *servlet* abstracto para configurar el *framework* para nuestra aplicación.
- Requiere de conocimientos de tecnologías web.

I JEE. Comparativa frameworks

	Servlets+JSP	Struts	Struts 2	JSF	JSF 2	ZK	ItsNat
MVC	√	√	√	√	√	√	√
MVVM						√	
AJAX			√ (*)		√	√	√
JCP estándar	√			√	√		
Page Centric	√	√	√	√	√		
SPI						√	√
Open Source	√	√	√	√	√	√(**)	√

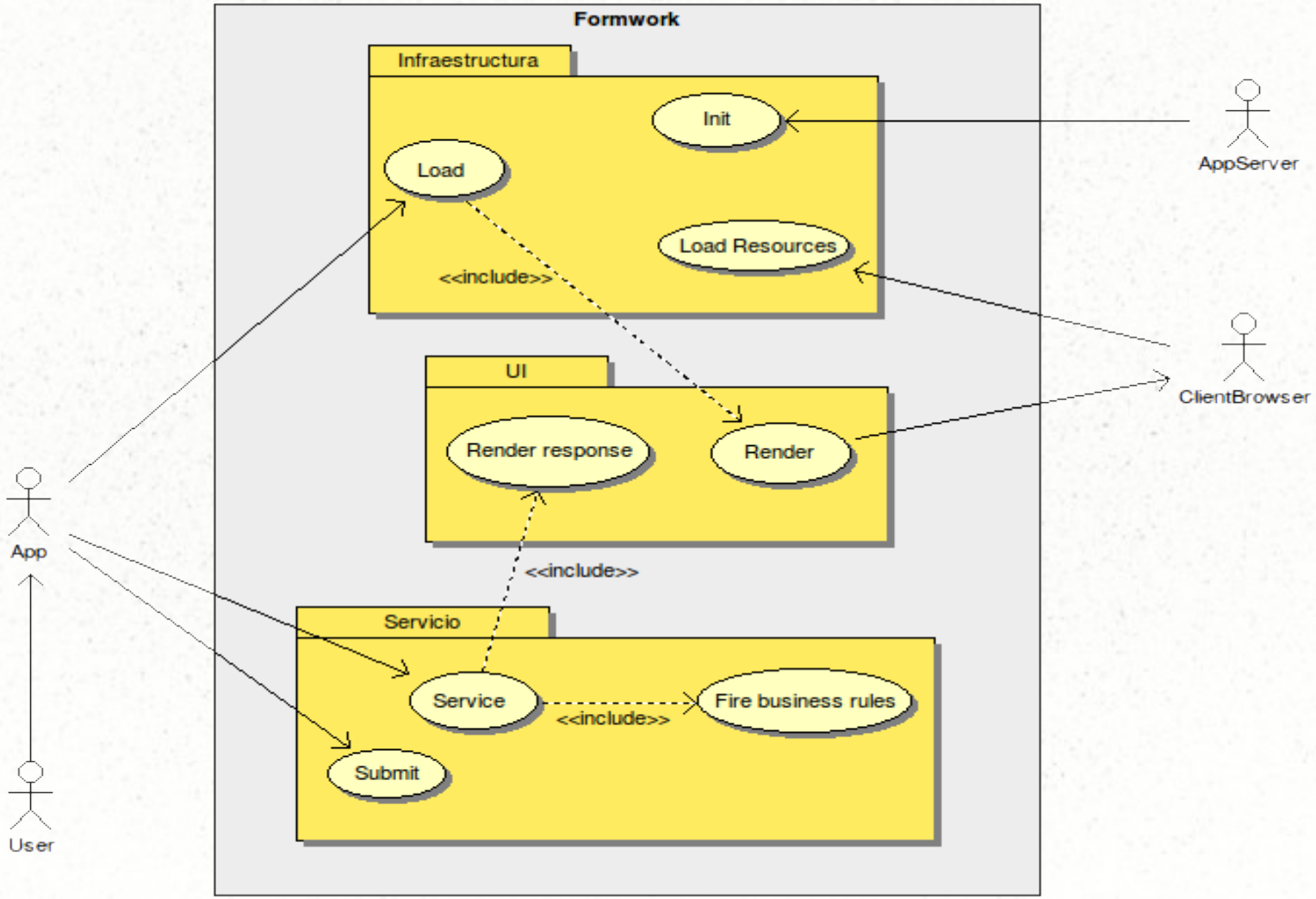
(*) Con el apoyo de DOJO Toolkit.

(**) Ediciones EE y PE sujetas a licencia comercial.

II Formwork. Requisitos

- Framework orientado a aplicaciones de pago de tributos simples.
- MVC
- Lenguaje marcas FWML para la creación de vistas.
- Aplicaciones formadas por una única página con un único formulario, formado por apartados y componentes.
- Controladores formados por clase que extienden la clase `GenericController`.
- AJAX mediante **jQuery**.
- Soporte a la ejecución de reglas de negocio mediante JBoss DROOLS.
- *Client + Server Fusion*.

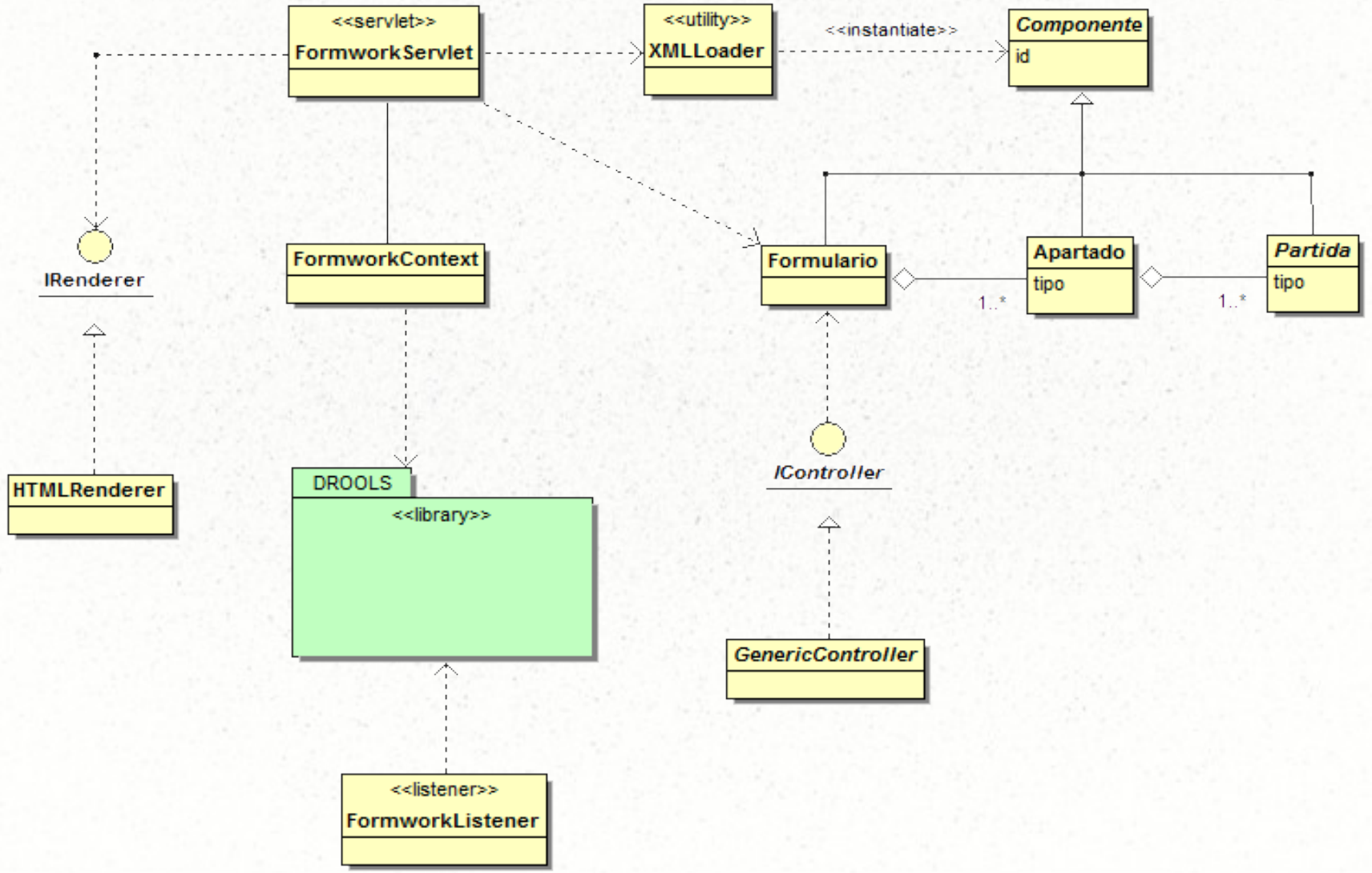
II Formwork. Análisis (y I)



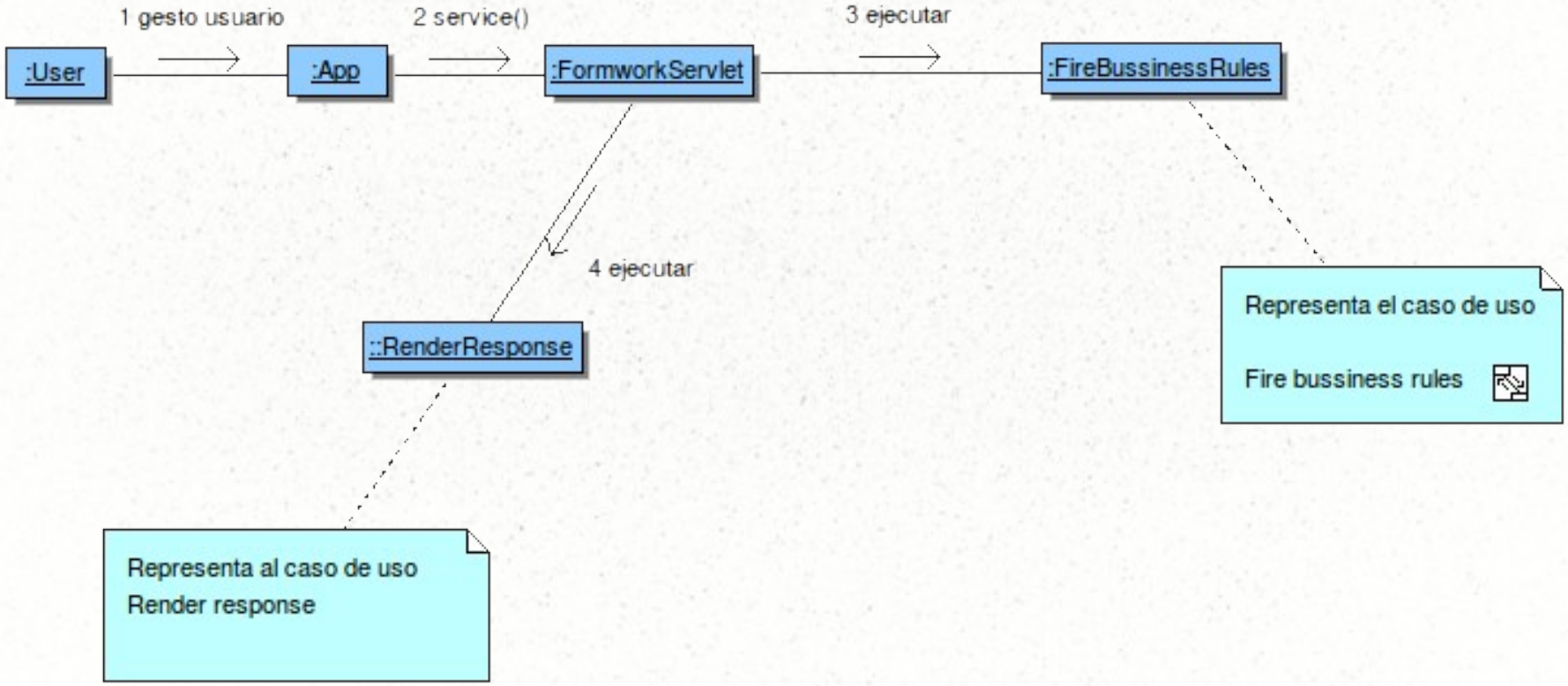
II Formwork. Análisis (y II)

Casos de uso	
Init	Inicialización *framework*.
Load	Carga de la página principal.
Load resurces	Carga de recusos de la página principal.
Render	Dibujo de la página principal.
Render response	Dibujo del resultado de una petición.
Service	Petición asíncrona desde el cliente.
Fire business rules	Ejecución de las reglas de negocio.
Submit	Presentación del formulario.
Actores	
User	Usuario de una aplicación `Formwork`.
App	Aplicación `Formwork`.
AppServer	Servidor de aplicaciones donde se despliega una aplicación `Formwork`.
ClientBrowser	Navegador del cliente.

II Formwork. Análisis (y III)



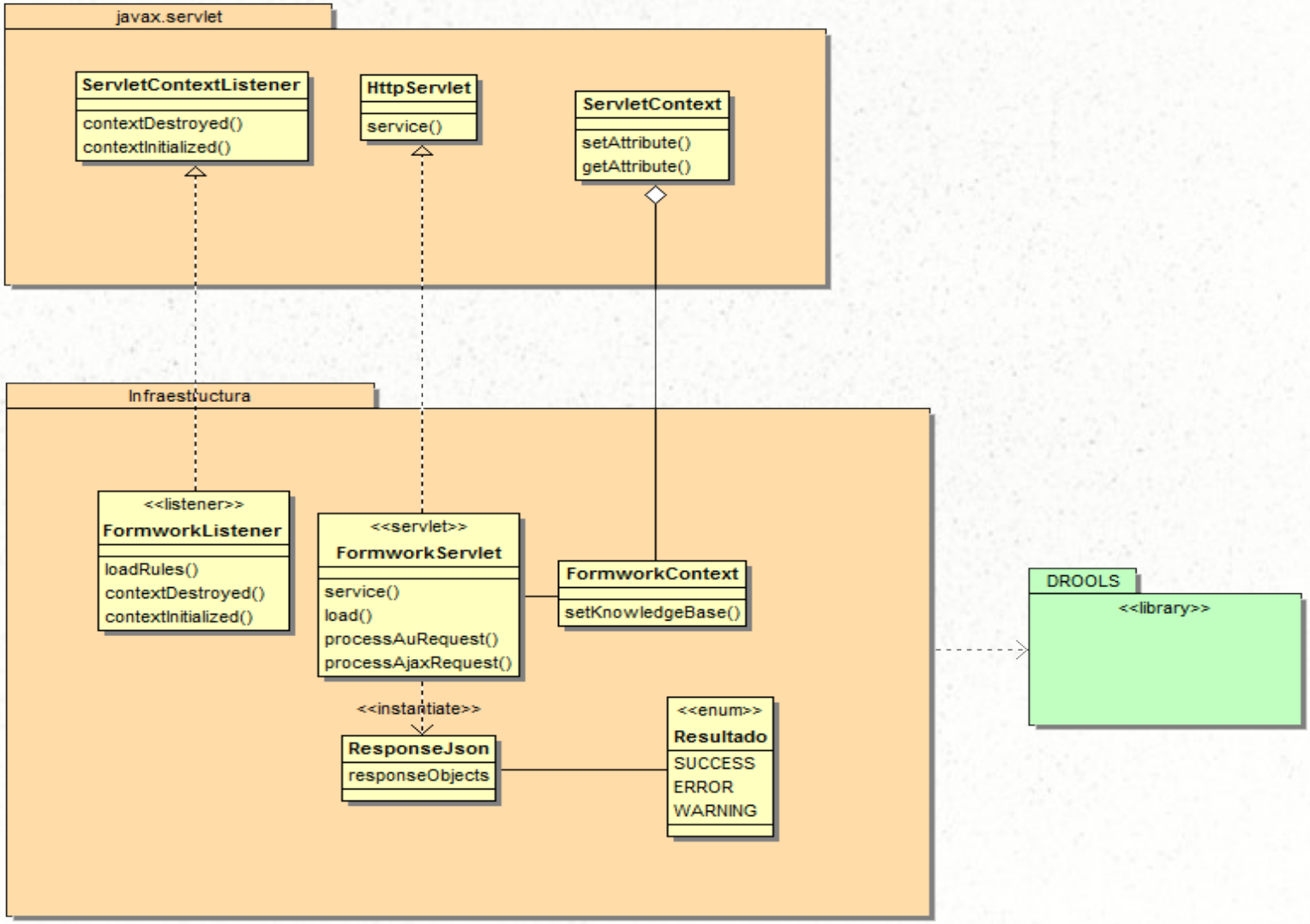
II Formwork. Análisis (y IV)



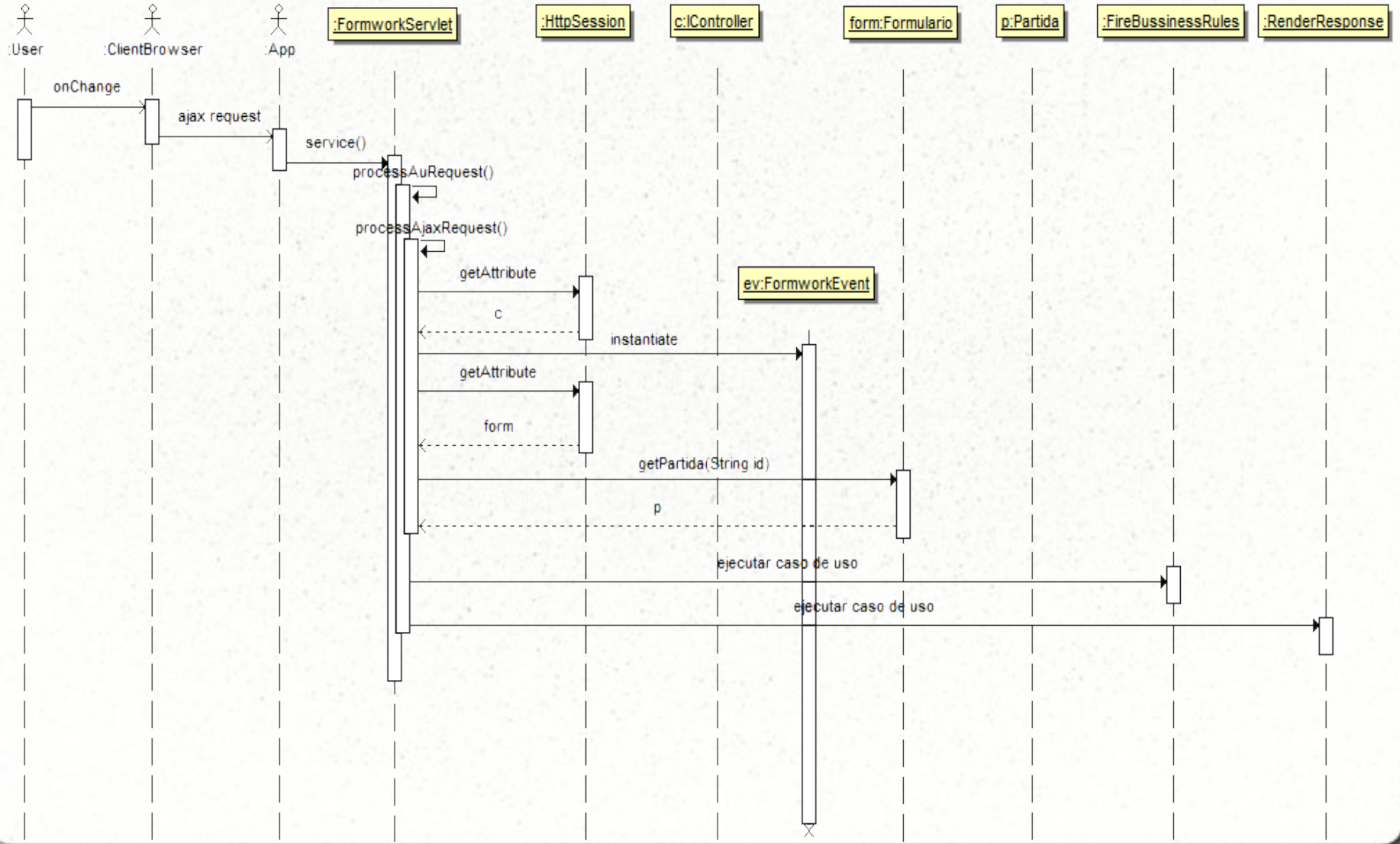
II Formwork. Análisis (y V)

- En las diapositivas anteriores hemos visto:
  - El diagrama de casos de uso
  - Una breve descripción de los casos de uso y de los actores implicados.
  - Un diagrama de clases de análisis
  - El diagrama de colaboración del caso de uso **Service**.
- En la memoria del PFC se puede estudiar con detalle el estudio del análisis de Formwork.

II Formwork. Diseño (y I)



II Formwork. Diseño (y II)



II Formwork. Diseño (y III)

- En las diapositivas anteriores hemos visto:
  - El diagrama de clases de diseño para el paquete infraestructura.
  - Un diagrama de secuencias para el caso de uso Service en un escenario de éxito.
- En la memoria del PFC se puede estudiar con detalle el estudio del diseño de Formwork.

II Formwork. Implementación (y I)

- Implementación en Java, JavaScript (con jQuery) y mvel (DROOLS):
  - Java 7 (OpenJDK y Oracle)
- El proyecto se ha gestionado con Apache Maven 3.
- Se ha usado git como SCV. Repositorio central en GitHub: `https://github.com/cachocenso/PFC-2012`
- IDE Eclipse 3.7 (Indigo) y 4.0.2 (Juno):
  - *Plugin* m2e.
  - *Plugin* Egit.
- Se ha desarrollado bajo:
  - Ubuntu 12.04 Precise Pangolin
  - Windows 7
  - Mac OSX 10.8.2 Mountain Lion

II Formwork. Implementación (y II)

- Las vistas se implementan con un lenguaje de marcas propio de Formwork llamado FWML.
- El FWML se procesa en el servidor donde se construye el árbol de componentes equivalente.
- El árbol de componentes se transforma en HTML con el apoyo de la librería Freemarker.
- El código HTML junto con los recursos (JavaScript, CSS etc) se envía al cliente.

```
<?xml version="1.0" encoding="UTF-8"?>

<formulario xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://www.uoc.edu/2012/fwp/formwork.xsd"
        id="m666"
        controlador="edu.uoc.pfc.formworktest.controller.TestController">

        <titulo>Modelo 666</titulo>
        <descripcion> Gravamen Especial Sobre Dos Conceptos Tributarios Que Me Acabo De Inventar. Modelo 666
        </descripcion>
        <apartado id="ap1" tipo="identificacion" contenido="nif,representante,nombre" />
        <apartado id="ap2" tipo="devengo"
                contenido="ejercicio, periodo(0A 1T 2T 3T 4T), fecha" />
```

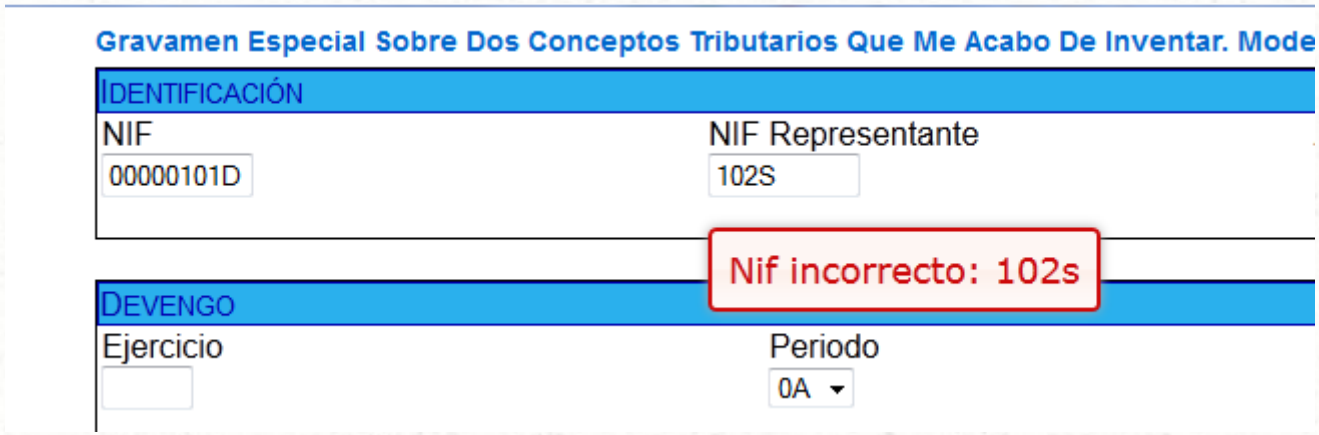
II Formwork. Implementación (y III)

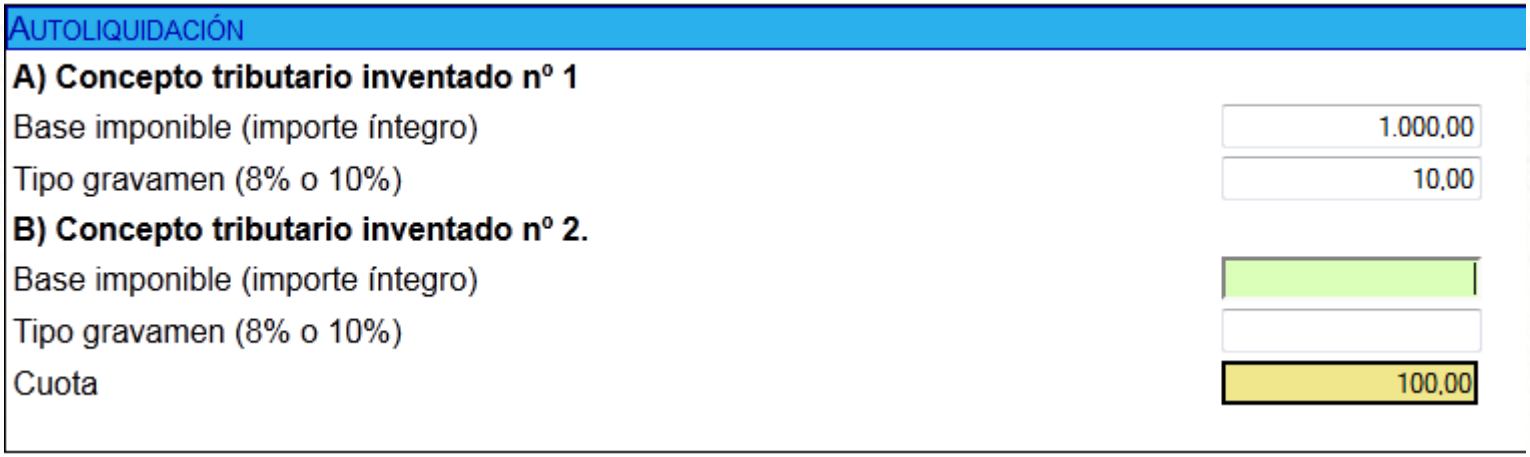
- `Init` y `Load` se implementan con un `ServletContextListener` y un s*ervlet* respectivamente:
  - `FormworkListener`
  - `FormworkServlet`
- Tanto el *listener* como el *servlet* se deben configurar en el descriptor de la aplicación `web.xml`.
- Una vez desplegada la aplicación, el *servlet* atenderá peticiones en dos URL distintas:
  - *.fwp : Carga de una página FWML
  - /au/* : Peticiones AJAX o de carga de recursos.

II Formwork. Implementación (y IV)

- En la página cliente se capturarán todos los eventos DOM onChange de todos los elementos `input` y `select` (con jQuery).
- En el gestor del evento se prepara una llamada AJAX al *servlet*. Se pasan como parámetros el id del componente cuyo valor ha cambiado y su nuevo valor.
- El *servlet* encapsula la información en un evento y se lo pasa al controlador de la aplicación para que ejecute las acciones que le correspondan (p.e. Ejecutar reglas de negocio con DROOLS).
- El servlet prepara la respuesta en formato JSON y la devuelve al cliente, que actualizará el árbol DOM en consecuencia con los resultados obtenidos.

II Formwork. Implementación (y V)





Conclusiones

- JEE es ya una tecnología madura para el desarrollo de aplicaciones empresariales.
- Parte de esa madurez se la han proporcionado la gran cantidad de *frameworks* que se han creado para cubrir todos los aspectos de la tecnología.
- El desarrollo de un nuevo *framework*, como el propuesto en este TFC, supone una buena oportunidad de profundizar en el conocimiento de las entrañas de JEE y de aplicar con éxito los conocimientos adquiridos durante la carrera