J2EE: frameworks de la capa de presentació

Melchor Vives Bernat
Enginyeria en Informàtica

Josep Maria Camps Riba
Consultor

Entrega PFC
20 de juny de 2011

Agraïments

Quan record el temps que he invertit els darrer anys, i no han estat pocs, en treurem assignatura a assignatura el cicle superior d'Enginyer en Informàtica sempre tinc present a la memòria la persona que més m'ha ajudat a aconseguir aquesta meta que em vaig imposar com un repta personal, aquesta persona és la meva parella Laura.

Vull agrair sobretot a na Laura la paciència que ha tingut amb jo al veure com la meva vida social es reduïa a fer PACs i pràctiques, fer feina i estudiar alhora no permeten disfrutar de massa temps lliure, i ella ha estat la persona que ha sofert més de prop els efectes adversos d'estudiar després de la feina.

Gràcies a ella amb els seus consells i els ànims continus he pogut sobrepassar els moments difícils en que la pèrdua de la fe i la constància perillaven assolir el meu objectiu personal.

Per altra banda, agrair també els meus pares que sempre m'han ajudat a continuar els estudis i han sigut molt comprensius en els moments més difícils. El suport constant també m'ha mantingut actiu durant els darrers anys d'estudi i per això els hi estic molt agraït.

No em puc oblidar de la UOC que m'ha permès treurem el cicle superior a distància ja que fent feina es quasi impossible de forma presencial.

També agrair als consultors i professors per la qualitat de les assignatures i la bona relació i comunicació que han demostrat en cada quatrimestre.

Finalment, voldria animar a totes les persones que com jo, dubtaven de la importància del cicle superior i les aconsello que ho intentin perquè una vegada arribat al final del camí i mirat enrera puc afirmar que ha valgut la pena.

Resum del PFC

El tema principal d'aquest PFC són els marcs de treball de la capa de presentació implementats amb Java EE per aplicacions web complexes.

Aquest PFC és un recorregut d'aquests marcs de treball començant per una recerca a Internet, seguint amb l'anàlisi dels marcs de treball més importants, l'obtenció de les característiques comunes a tots ells, el disseny i implementació d'un marc de treball propi i finalment, crear una aplicació d'exemple que empri el nostre marc de treball.

En la primera part del projecte, es realitzarà una explicació i avaluació del mateix, així com una planificació del pla del projecte que inclourà les tres etapes principals, que son: l'estudi dels marcs de treball, el disseny del nostre marc de treball i la creació d'una aplicació d'exemple que utilitzi el nostre marc de treball.

El primer punt important del projecte serà entendre que és un mar de treball, per què tenim marcs de treball orientats a la capa de presentació, una introducció al llenguatge Java i a la tecnologia Java EE. Després, començarem una recerca per Internet per identificar els principals marcs de treball Java EE per la capa de presentació, els analitzarem i agruparem en diversos grups segons les seves característiques. Finalment, extraurem les característiques principals que hauria de tenir qualsevol marc de treball Java EE per la capa de presentació.

El segon punt important del projecte es pot resumir en la creació d'un marc de treball propi per la capa de presentació. Com a punt de partida, s'hauran d'implementar les característiques que hauria de tenir qualsevol marc de treball obtingudes al punt anterior. S'expondrà en què consisteix el nou marc de treball, un anàlisis i disseny seguit de la seva codificació per acabar amb un tutorial de com s'ha d'utilitzar.

El tercer i últim punt important consisteix en la creació d'una aplicació web a mode d'exemple d'utilització del nostre marc de treball i com aquest ens ajuda en la codificació del projecte.

Aquest PFC exposa les idees principals dels marcs de treball de la capa de presentació i gràcies a la creació i utilització d'un marc de treball propi ens apropa a les característiques i problemes principals que intenten resoldre.

Índex

Memòria

1.1 Etapa del pla de treball

1.1.1 Context del PFC

1.1.1.1 J2EE

Actualment al mon empresarial es quasi imprescindible l'ús de l'informàtica per suportar els processos essencials del negoci i tenir una avantatge competitiva sobre les altres empreses. Hi ha moltes solucions per implantar aquesta nova tecnologia, però una de les més esteses i conegudes es la J2EE.

J2EE proporciona a les empreses un conjunt d'eines, llenguatges i l'infrastructura necessària per desenvolupar i explotar totes les aplicacions que ajudin a mantenir i millorar els productes o els processos que realitzin aquestes empreses.

Aquesta tecnologia es propietat de SUN i empra majoritàriament el llenguatge JAVA. J2EE ofereix un conjunt d'especificacions que altres empreses poden implementar per generar nous llenguatges, aplicacions, utilitats, servidors, etc.

1.1.1.2 Marcs de treball

Una vegada creat l'estàndard J2EE, posteriorment s'han creat una multitud de "frameworks", que a partir d'ara anomenarem marcs de treball, i que s'han inspirat en agrupar una sèrie de característiques que ajuden a desenvolupar millor les aplicacions dissenyades amb J2EE.

Aleshores, un marc de treball no es més que a partir d'una especificació base, en aquest cas J2EE, una agrupació de bones pràctiques, de codi reutilitzable, de l'implementació de patrons de disseny per solucionar problemes recurrents, d'unificar nomenclatures, de llibreries amb funcionalitats ja especificades, d'abstracció entre varies capes, d'agrupar altres marcs de treball, de proporcionar més potència, flexibilitat, etc.

En conclusió, un marc de treball està dissenyat per ajudar a millorar les aplicacions construïdes amb la base comú del dos.

1.1.1.3 Marcs de treball de la capa de presentació

Els marcs de treball solen estar associats a alguna de les capes en les que es pot dividir un projecte informàtic. Tenim marcs de treball que només apliquen a la vista de la capa de presentació, d'altres que donen solució al patró MVC de la capa de presentació, altres també inclouen la capa de negoci, també en tenim per agilitar i mapejar la capa de persistència i altres que serveixen per generar serveis per integrar aplicacions de diverses organitzacions.

En el nostre cas, els marcs de treball que més ens interessen són els relacionats amb la capa de presentació, descartant els que només apliquen a la vista, es a dir, marcs de treball desenvolupats amb JavaScript per millorar l'interacció amb l'usuari, o les anomenades aplicacions RIA que incorporen components per emular les típiques aplicacions d'escriptori, igualment estan desenvolupades amb JavaScript i l'intercanvi d'informació es fa purament amb cridades AJAX.

Així doncs, ens centrarem amb els marcs de treball que incorporen el patró MVC (model vista controlador) i impliquen tant una part del client com una altra del servidor. Dins aquests marcs de treball s'inclou des del codi que s'executa dins el navegador, les peticions que es realitzen al servidor, l'intercanvi de dades de la sessió, el control d'aquestes peticions, execució de la funcionalitat adequada, aplicació del model, redirecció a la vista adequada i altres característiques associades a aquest flux.

No es tindran en compte les característiques associades a les capes de negoci, de persistència, de serveis web i altres possibles combinacions. Afegim aquesta puntualització perquè no tots els marcs de treball estan incloses dins una única capa, per exemple, els marcs de treball que inclouen l'especificació de la tecnologia EJB (Enterprise JavaBean), ofereixen solucions per les capes de negoci i persistència.

1.1.1.4 Aplicacions que empren aquests frameworks

Durant la darrera dècada s'ha popularitzat la web com el suport principal per interactuar amb les aplicacions, aquest fet es degut a l'accessibilitat arreu del mon d'Internet, la millora de les infrastructures de comunicació, la flexibilitat que ofereix, la d'escalabilitat i l'avantatge de ser multi-plataforma.

Tota aplicació web (o d'escriptori) inclou una capa de presentació, la qual té molt de pes dins l'arquitectura del projecte ja que una part important del codi es correspon amb esta capa, tenim les vistes, les controladores, els models a més de varis fitxers de configuració.

Aquests marcs de treball ofereixen abstracció sobre les parts més importants d'aquesta capa, una conjunt de guies de com desenvolupar cada part i funcionalitat, normalització dels estàndards de nomenclatura i en general homogeneïtzar el desenvolupament de les aplicacions amb avantatges proporcionades per el marc de treball.

1.1.1.5 Motivacions personals

Totes les capes en que es pot dividir una aplicació són importants, però possiblement la capa de presentació és la més completa, complexa i que engloba més característiques de les aplicacions web.

Totes aquestes argumentacions són les que m'han dut a seleccionar aquesta proposta per el PFC. Si cerquem per Internet pareix que la tendència és la mateixa, es poden trobar moltíssims marcs de treball enfocats a la capa de presentació que ofereixen prestacions semblants però la seva dimensió, rendiment i implementació són molt diferents.

La recerca d'aquests marcs de treball, el seu estudi posterior, la recerca de les seves característiques bàsiques comunes i la comparació del diferents marcs de treball conformen una part molt interessant d'aquesta memòria.

Una altre punt que m'agradaria mencionar es l'utilització del marc de treball adequat segons l'empresa que el desenvolupi, l'experiència dels recursos del projecte, les necessitats del client, l'infrastructura existent, etc. Seleccionar el marc de treball adequat per el projecte i el client és una decisió molt important que pot influir en el temps d'implementació del projecte i les expectatives de d'execució i rendiment esperades per els usuaris de l'aplicació.

1.1.1.6 Aportacions de la comunitat

Degut a l'importància de la capa de presentació, cada vegada sorgeixen nous marcs de treballs, això no vol dir que el darrer sigui el millor, també podríem pensar que ho son els primers perquè han tingut més temps per resoldre els seus bugs i hagin pogut evolucionar les característiques més emprades per la comunitat.

Cada marc de treball té un abast diferent, està dissenyat per resoldre unes abstraccions, suportar un rendiment, arribar a un nivell de qualitat, etc. Així doncs, el més important és seleccionar el marc de treball que s'adapti millor a les necessitats del projecte, els recursos i el client.

1.1.2 Objectius del PFC

1.1.2.1 Objectius generals de J2EE

J2EE és una tecnologia molt madura que s'ha instaurat al mon informàtic amb molta força i robustesa, és internacionalment reconeguda per les avantatges que proporciona en la construcció d'aplicacions software com en l'explotació de les mateixes.

Des dels inicis del llenguatge Java, la seva especificació de l'estàndard J2SE i J2EE, entre altres, ha anat madurant i adaptant-se a les necessitats del mercat, de les empreses i finalment dels usuaris que han d'emprar aquestes aplicacions.

J2EE aporta eines per desenvolupar tota classe d'aplicacions, ja siguin simples pàgines web, complexes backoffice, web services, aplicacions amb components distribuïts, portals comercials, intranets, etc.

A més, gràcies a la JVM (Java Vitural Machine) totes les aplicacions J2EE són multi-plataforma, el software amb llenguatge Java es compilat i sa generen uns fitxers anomenats bytecode amb extensió .class que la JVM executa amb independència del sistema operatiu i del hardware on està instal·lat.

Així doncs, J2EE és una solució elegant i completa per a totes les necessitats que puguin anar sorgint tant en el mon empresarial, universitari, nacional, particular, etc. Aprendre aquesta tecnologia ens aporta avantatges en el moment de cercar feina o competir per un lloc de treball millor.

1.1.2.2 Objectius específics de la capa de presentació

Una vegada hem comprovat i confirmat que J2EE ens ajuda a desenvolupar millor software ens centrarem en les capes en que es pot dividir una aplicació: presentació, negoci, persistència, serveis web, etc.

S'ha seleccionat la capa de presentació per ser estudiada dins aquest PFC, primer es realitzarà un estudi dels marcs de treball que engloben aquesta capa i ens ajuden amb el seu anàlisis, disseny, desenvolupament, proves i manteniment; després es crearà un nou marc de treball a partir de les característiques més importants que ha de tenir un marc de treball dedicat a la capa de presentació; i finalment crear una aplicació d'exemple amb l'ajuda dels marc de treball que haguem dissenyat.

Els darrers anys, s'ha popularitzat l'implementació del patró de disseny MVC (model vista controlador) com la millor solució per dissenyar la separació de l'abstracció de les característiques úniques de cada un d'aquests punts. Analitzar com els marcs de treballs actuals implementen aquesta solució conformarà una part important del primer bloc del PFC.

La capa de presentació té molt de pes dins les aplicacions, és l'interfície entre l'usuari i el software, té una complexitat considerable de la qual derivaran un conjunts de característiques que els marcs de treball han de suportar.

Dins aquesta subcapa també s'inclou la vista, on es generarà el codi interpretat per els navegadors web; també tenim la subcapa controladora que gestionarà el tràfic de les peticions HTTP del usuaris; finalment amb la subcapa model es construirà l'estructura de les vistes i l'informació que hi ha d'aparèixer.

Quan un usuari està navegant per una web, sigui del tipus que sigui, es produeix un intercanvi d'informació mitjançat peticions http des del navegador al servidor i viceversa. Aquest intercanvi te bàsicament dues parts, la capçalera on s'indica la direcció de la petició i el cos on s'envia un formulari amb l'informació necessària per la lògica que s'executa al servidor. Els marcs de treball també ofereixen suport per integrar de forma consistent tots aquests punts, des de com es mostrarà l'informació al client, passant per la lectura del camps enviats amb els formularis al servidor, com mantenir totes aquestes dades vives en la sessió que mantenen el client i el servidor.

1.1.3 Enfocament i mètode seguit

El projecte s'ha dividir en tres grans blocs: en el primer s'avaluaran els marcs de treball existents actualment i que engloben la capa de presentació en aplicacions J2EE, s'estudiaran les característiques comunes i es relacionaran entre si; en el segon bloc es crearà un nou marc de treball a partir dels punts més importants que hem abstret de l'anàlisi anterior, aportant les nostres pròpies solucions segons els requeriments que ens haguem marcat; finalment crearem una petita aplicació per il·lustrar l'utilització del nostre marc de treball.

Amb l'estudi dels marcs de treball actuals volem conèixer l'estat del sector, averiguar quins sols el marcs de treball més coneguts i populars, els que tenen més antiguitat, els més complets i els mes àgils, i de tots ells estudiar els seus punts forts i febles, llistar les seves característiques i comparar-los entre ells, relacionar cada marc de treball amb un tipus d'organització i projecte, assignar un grau de complexitat a l'aprenentatge, desenvolupament i manteniment de cada un d'ells, i seleccionar el més adequat a les nostres necessitats.

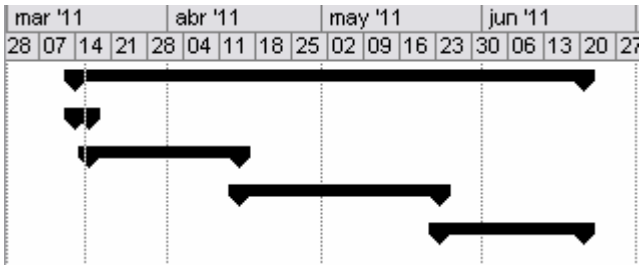
Per la creació del nostre marc de treball haurem de decidir el seu abast, per quin tipus d'aplicacions va dirigit, tots els recursos que podrem emprar per dissenyar-lo i en general tot el relacionat amb dissenyar la nostra pròpia solució. Una cop s'hagin formulat les bases del nostre marc de treball haurem de començar a dissenyar seguint una especificació ben acotada i dirigida al nostre objectiu. Per aconseguir una solució de qualitat emprarem totes les eines disponibles com l'ús de diagrames UML tant per els diagrames de classes, casos d'ús, diagrames d'interacció, aplicació de patrons de disseny, etc. Una vegada dissenyat, especificat i implementat el nostre marc de treball, només faltarà redactar la documentació necessària per utilitzar-lo.

Finalment, crearem una petita aplicació, a mode de tutorial, per exposar l'ús d'aquest marc de treball, i demostrar com les seves avantatges ens ajuden en el desenvolupament ràpid, en ordenar i simplificar el codi font, en estructurar el projecte, el unificar nomenclatura, en agilitzar el manteniment posterior, etc. Aquesta aplicació ens servirà de guia per exhibir tots els avantatges dels marcs de treball de la capa de presentació, i en particular, del nostre disseny.

1.1.4 Planificació del projecte.

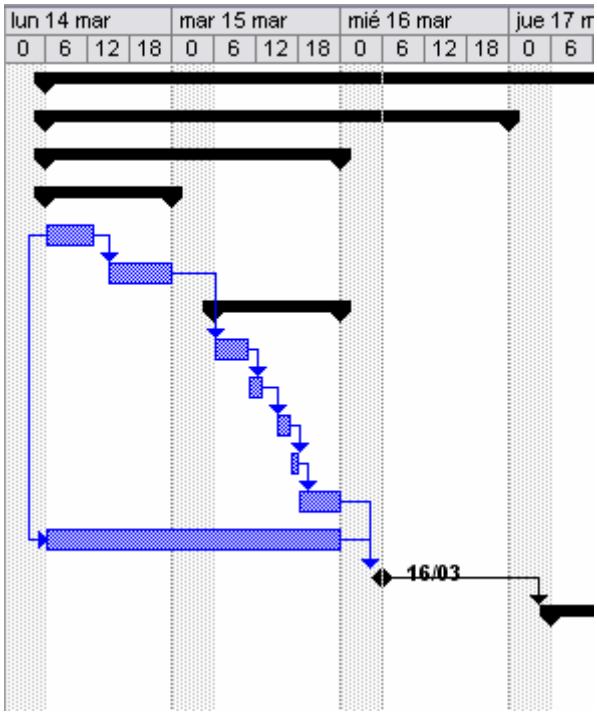
1.1.4.1 Planificació global de projecte

Nombre de tarea	Duración	Comienzo	Fin
− PFC de J2EE sobre la capa de presentació	71 días?	lun 14/03/11	lun 20/06/11
+ Etapa del pla de treball	3 días?	lun 14/03/11	mié 16/03/11
+ Etapa d'anàlisis del frameworks	21 días?	jue 17/03/11	jue 14/04/11
+ Etapa d'especificació i implementació del framework	27 días?	vie 15/04/11	lun 23/05/11
+ Etapa de creació d'una aplicació d'exemple	20 días?	mar 24/05/11	lun 20/06/11



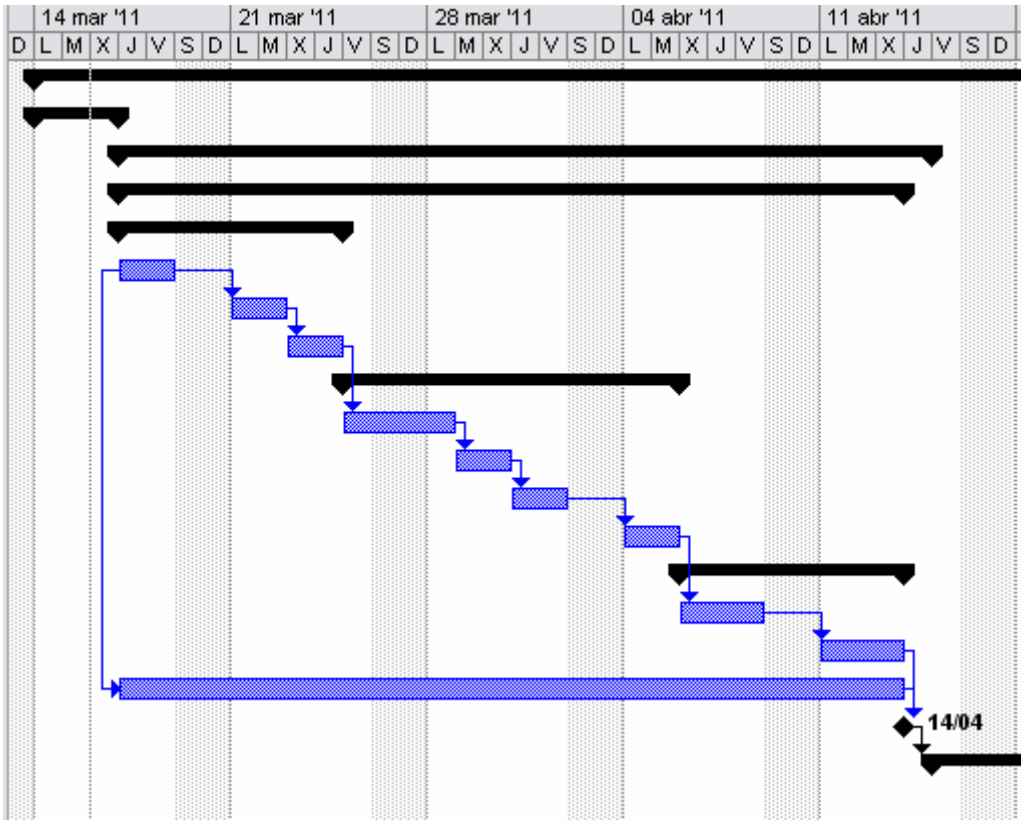
1.1.4.2 Planificació de l'etapa del pla de treball

Nombre de tarea	Duración	Comienzo	Fin
– PFC de J2EE sobre la capa de presentació	**71 días?**	**lun 14/03/11**	**lun 20/06/11**
– Etapa del pla de treball	**3 días?**	**lun 14/03/11**	**mié 16/03/11**
– Tasques per generar la PAC1	**2 días**	**lun 14/03/11**	**mar 15/03/11**
– Planificació temporal	**1 día**	**lun 14/03/11**	**lun 14/03/11**
Crear el top-down de la planificació temporal al	4 horas	lun 14/03/11	lun 14/03/11
Crear la planificació temporal amb Microsoft Pro	4 horas	lun 14/03/11	lun 14/03/11
– Crear el document per la memòria	**1 día**	**mar 15/03/11**	**mar 15/03/11**
Incloure la portada (seguint la plantilla)	2 horas	mar 15/03/11	mar 15/03/11
Incloure l'estructura (seguint la plantilla)	2 horas	mar 15/03/11	mar 15/03/11
Incloure el pla de treball (seguint la plantilla)	2 horas	mar 15/03/11	mar 15/03/11
Editar la descripció del PFC	1 hora	mar 15/03/11	mar 15/03/11
Editar els objectius generals i específics del PF	1 hora	mar 15/03/11	mar 15/03/11
Seguiment actualització de la memòria	2 días?	lun 14/03/11	mar 15/03/11
Fita entrega de la PAC1	1 día?	mié 16/03/11	mié 16/03/11
+ Etapa d'anàlisis del frameworks	**21 días?**	**jue 17/03/11**	**jue 14/04/11**
+ Etapa d'especificació i implementació del framework	**27 días?**	**vie 15/04/11**	**lun 23/05/11**
+ Etapa de creació d'una aplicació d'exemple	**20 días?**	**mar 24/05/11**	**lun 20/06/11**



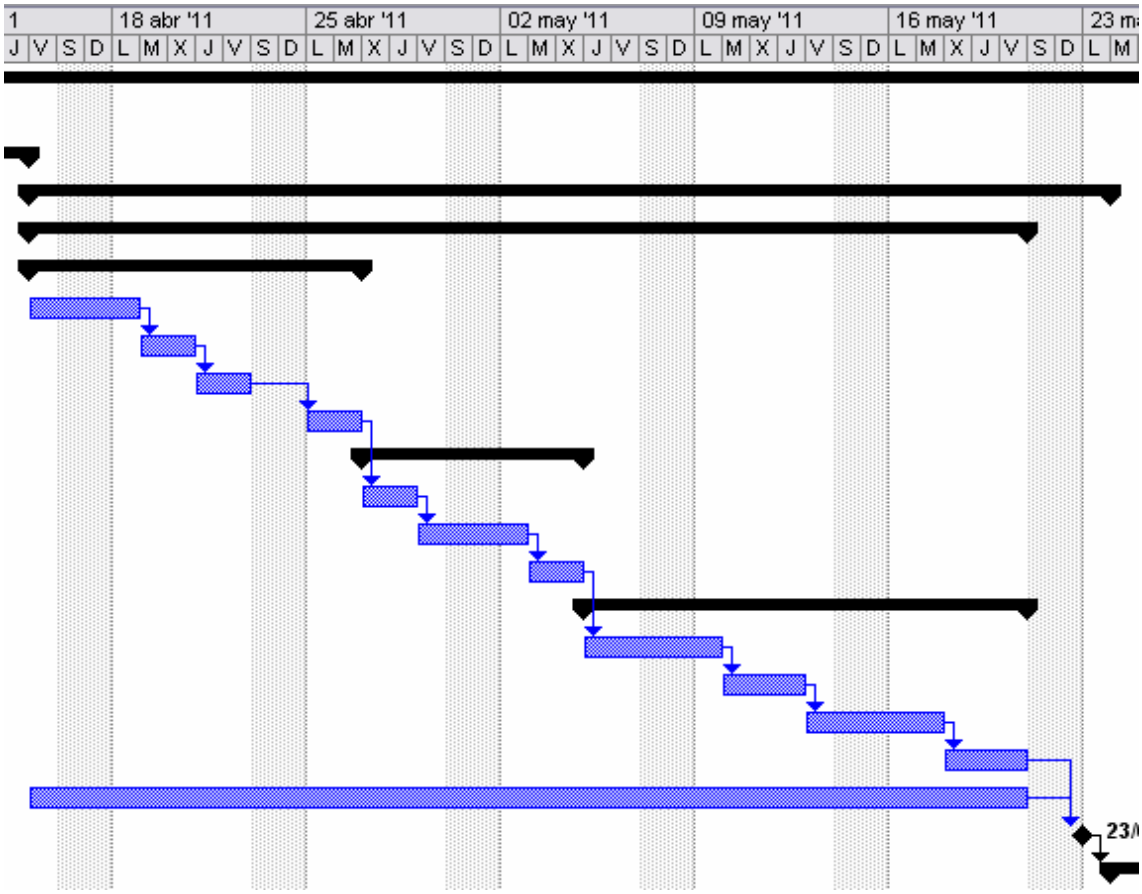
1.1.4.3 Planificació de l'etapa d'anàlisis del frameworks

Nombre de tarea	Duración	Comienzo	Fin
− PFC de J2EE sobre la capa de presentació	**71 días?**	**lun 14/03/11**	**lun 20/06/11**
+ Etapa del pla de treball	**3 días?**	**lun 14/03/11**	**mié 16/03/11**
− Etapa d'anàlisis del frameworks	**21 días?**	**jue 17/03/11**	**jue 14/04/11**
− Tasques per generar la PAC2	**20 días**	**jue 17/03/11**	**mié 13/04/11**
− Anàlisis de la situació actual	**6 días**	**jue 17/03/11**	**jue 24/03/11**
Cercar per Internet frameworks J2EE que inclo	2 días	jue 17/03/11	vie 18/03/11
Ordenar-los/prioritzarlos per popularitat i ús	2 días	lun 21/03/11	mar 22/03/11
Analitzar les seves característiques, funcionali	2 días	mié 23/03/11	jue 24/03/11
− Valoracions personals	**8 días**	**vie 25/03/11**	**mar 05/04/11**
Relacionar les característiques/funcionalitats e	2 días	vie 25/03/11	lun 28/03/11
Crear una llista dels punts més importants (que	2 días	mar 29/03/11	mié 30/03/11
Relacionar aquest punts amb els problemes ge	2 días	jue 31/03/11	vie 01/04/11
Relacionar aquests problemes amb patrons de	2 días	lun 04/04/11	mar 05/04/11
− Patró MVC	**6 días**	**mié 06/04/11**	**mié 13/04/11**
Introducció/explicació del patró MVC	3 días	mié 06/04/11	vie 08/04/11
Relacionar el MVC amb la solució aportada per	3 días	lun 11/04/11	mié 13/04/11
Seguiment actualització de la memòria	20 días?	jue 17/03/11	mié 13/04/11
Fita entrega de la PAC2	1 día?	jue 14/04/11	jue 14/04/11
+ Etapa d'especificació i implementació del framework	**27 días?**	**vie 15/04/11**	**lun 23/05/11**
+ Etapa de creació d'una aplicació d'exemple	**20 días?**	**mar 24/05/11**	**lun 20/06/11**



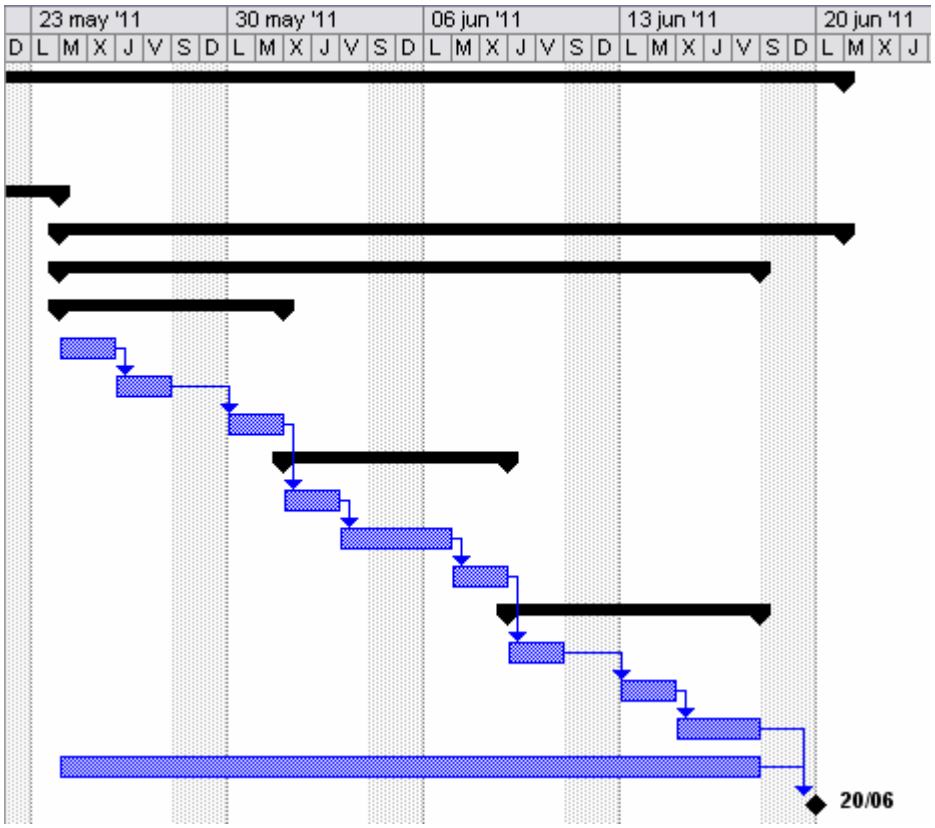
1.1.4.4 Planificació de l'etapa d'especificació i implementació del framework

Nombre de tarea	Duración	Comienzo	Fin
- PFC de J2EE sobre la capa de presentació	**71 días?**	**lun 14/03/11**	**lun 20/06/11**
+ Etapa del pla de treball	**3 días?**	**lun 14/03/11**	**mié 16/03/11**
+ Etapa d'anàlisis del frameworks	**21 días?**	**jue 17/03/11**	**jue 14/04/11**
- Etapa d'especificació i implementació del framework	**27 días?**	**vie 15/04/11**	**lun 23/05/11**
- Tasques per generar la PAC3	**26 días**	**vie 15/04/11**	**vie 20/05/11**
- Anàlisis del nou framework	**8 días**	**vie 15/04/11**	**mar 26/04/11**
Abast del framework	2 días	vie 15/04/11	lun 18/04/11
Comparació amb els frameworks actuals	2 días	mar 19/04/11	mié 20/04/11
Llistat de requeriments funcionals/estructurals	2 días	jue 21/04/11	vie 22/04/11
Llistat de casos d'ús (suportats per el framwor	2 días	lun 25/04/11	mar 26/04/11
- Disseny del nou framework	**6 días**	**mié 27/04/11**	**mié 04/05/11**
Diagrama de classes (UML)	2 días	mié 27/04/11	jue 28/04/11
Patrons de disseny (emprats)	2 días	vie 29/04/11	lun 02/05/11
Aplicació del MVC	2 días	mar 03/05/11	mié 04/05/11
- Implementació del nou framework	**12 días**	**jue 05/05/11**	**vie 20/05/11**
Crear l'estructura del framework	3 días	jue 05/05/11	lun 09/05/11
Crear les classes interface i classes abstracte	3 días	mar 10/05/11	jue 12/05/11
Crear els properties, xmls, per configurar el fra	3 días	vie 13/05/11	mar 17/05/11
Implementar el framework i deixar-lo llest per s	3 días	mié 18/05/11	vie 20/05/11
Seguiment actualització de la memòria	26 días?	vie 15/04/11	vie 20/05/11
Fita entrega de la PAC3	1 día?	lun 23/05/11	lun 23/05/11
+ Etapa de creació d'una aplicació d'exemple	**20 días?**	**mar 24/05/11**	**lun 20/06/11**



1.1.4.5 Planificació de l'etapa de creació d'una aplicació d'exemple

Nombre de tarea	Duración	Comienzo	Fin
- PFC de J2EE sobre la capa de presentació	**71 días?**	**lun 14/03/11**	**lun 20/06/11**
+ Etapa del pla de treball	**3 días?**	**lun 14/03/11**	**mié 16/03/11**
+ Etapa d'anàlisis del frameworks	**21 días?**	**jue 17/03/11**	**jue 14/04/11**
+ Etapa d'especificació i implementació del framework	**27 días?**	**vie 15/04/11**	**lun 23/05/11**
- Etapa de creació d'una aplicació d'exemple	**20 días?**	**mar 24/05/11**	**lun 20/06/11**
- Tasques per generar la PAC4	**19 días**	**mar 24/05/11**	**vie 17/06/11**
- Anàlisis de l'exemple	**6 días**	**mar 24/05/11**	**mar 31/05/11**
Explicació de l'aplicació d'exemple	2 días	mar 24/05/11	mié 25/05/11
Llistat de requeriments funcionals	2 días	jue 26/05/11	vie 27/05/11
Llistat de casos d'ús	2 días	lun 30/05/11	mar 31/05/11
- Disseny de l'exemple	**6 días**	**mié 01/06/11**	**mié 08/06/11**
Diagrama del model entitat relació	2 días	mié 01/06/11	jue 02/06/11
Diagrama del flux de navegació de les peticions	2 días	vie 03/06/11	lun 06/06/11
Diagrama de les capes de l'aplicació	2 días	mar 07/06/11	mié 08/06/11
- Implementació	**7 días**	**jue 09/06/11**	**vie 17/06/11**
Codificar les classes de l'aplicació emprant el f	2 días	jue 09/06/11	vie 10/06/11
Provar l'aplicació: funcionalitats, requeriments,	2 días	lun 13/06/11	mar 14/06/11
Generació d'un WAR/EAR per entregar	3 días	mié 15/06/11	vie 17/06/11
Seguiment actualització de la memòria	19 días?	mar 24/05/11	vie 17/06/11
Fita entrega de la PAC4 que es correspon amb la memòria	1 día?	lun 20/06/11	lun 20/06/11



1.2 Etapa d'anàlisis dels marcs de treball

1.2.1 Anàlisis de la situació actual

1.2.1.1 Introducció

Des de l'aparició del llenguatge Java, desenvolupat per Sun Microsystems, l'any 1995, la seva popularitat s'ha incrementat any rera any. Actualment, està globalment estes i és emprat en la majoria d'empreses, tant per les aplicacions internes, com les web, com per productes comercials.

Java juntament amb .Net i PHP, són les tres tecnologies més populars d'Internet, tant pe nombre d'usuaris, per desenvolupadors com per les empreses que els empren.

Durant aquesta darrera època la competitivitat ha propiciat l'aparició dels anomenats "frameworks", o marcs de treball, degut a la dura competència, que durant els anys s'ha anat incrementat amb volum i en qualitat. Per aquest fet, les empreses han hagut de cercar solucions per abaratir el cost dels projectes informàtics, disminuir el temps de les entregues, millorar la qualitat del producte, aconseguir aplicacions més mantenibles, reutilitzar components ja existents, etc.

Els marcs de treball aporten molts de beneficis a tots els nivells: empreses, aplicacions, desenvolupadors, caps de projecte, etc. Cada marc de treball té uns objectius, finalitats, característiques, complexitat, qualitat del producte, etc. Així doncs, seleccionar el millor marc de treball per l'empresa/departament/equip/desenvolupadors/client/aplicació és una de les decisions més importants ja que sinó es fa correctament por produir l'efecte contrari.

Un dels primers marcs de treball Java va ser Struts. Va ser creat per Craig R. McClanahan i es va incloure al projecte Jakarta d'Apache l'any 2000. Aquest marc de treball tenia com a objectiu donar suport al patró de disseny model-vista-controlador (MVC). Al llarg dels anys 'ha anat evolucionant fins que l'any 2008 va sortir Struts2, combinació de Struts1 i un altre marc de treball d'Apache anomenat WebWork, una versió molt més moderna que podia competir amb els nous marcs de treball més nous i adaptats a les noves tendències de programació.

1.2.1.2 Marcs de treball a Internet

Fent una recerca per Internet es pot trobar una multitud de marcs de treball Java.

Cada un d'aquests marcs de treball, presentats a continuació, està basat en Java EE en general i alguns amb alguna nova especificació del mateix, com JSF.

Hi ha marcs de treball que només estan pensats per donar suport a la capa de presentació com per exemple RichFaces i altres donen suport a totes les capes com Seam.

Cada marc de treball té una motivació única i un objectiu final ben definit, però a més, tots comparteixen les característiques bàsiques dels marcs de treball com per exemple, proporcionar les eines necessàries per desenvolupar molt ràpidament accions molt

repetitives.

A continuació presentem un llistat del marcs de treballs trobats per Internet.

- ADF
- Aurora
- Beehive
- Dinámica
- Echo3
- Facelets
- GWT
- Grails
- ICEfaces
- JSF
- MyFaces
- Play!
- PrimeFaces
- RichFaces
- Seam
- Spring MVC
- Stripes
- Struts
- Tapestry
- Turbine
- Velocity
- WebObjects
- Wichet
- ZK
- jQuery4jsf

1.2.1.3 Agrupació dels marcs de treball

Per abordar les característiques particulars de cada un d'aquests marcs de treball, el millor serà agrupar-los. Tot marc de treball té les característiques d'un marc de treball, del seu grup i d'ell mateix, aquestes darreres seran les diferències entre els seus companys de grup.

Així doncs, partirem de la base que tots aquest marcs de treball estan creats amb Java EE i donen suport a la capa de presentació.

1. MVC

En el primer grup s'han inclòs els marcs de treball basats només amb Java EE i amb l'implementació del model-vista-controlador MVC.

- Struts
- Spring MVC

- Seam
- Tapestry
- Wichet
- Turbine
- Stripes

2. JSF

L'any 2004 va sortir una nova especificació basada en Java anomenada Java Server Faces (JSF) per donar més potencia a les interfícies d'usuari proporcionar components Java amb estat, events, validacions, etc.. L'any 2009 va sortir la darrera versió, la 2.0.

- JSF (Reference Implementation de Sun Microsystems)
- RichFaces (JBoss Community)
- ICEfaces
- PrimeFaces
- MyFaces (Apache Software Foundation)
- jQuery4jsf
- Facelets

3. Java-Ajax-RIA

La simplicitat de les aplicacions ha propiciat l'aparició de marcs de treball que volen emular els GUI i comportaments de les conegudes aplicacions d'escriptori. Bàsicament, hi ha dues modalitats: els marcs de treball escrit només amb JavaScript+Ajax (que no tindrem en compte) i els marcs de treball que es programen amb Java però generen codi JavaScript+Ajax per interactuar amb el servidor, a continuació en veurem uns quants.

- Oracle ADF
- ZK
- GWT
- Echo3 Web Framework

4. Altres

Finalment, també tindrem en compte, els següents marcs de treball, que per les seves característiques tant heterogènies els hem agrupat en aquest grup.

- Grails
- Play!
- Aurora
- Beehive
- Dinámica
- Velocity

1.2.1.4 Presentació d'alguns marcs de treball

A continuació veurem dos marcs de treball de cada grup. Els he seleccionat per la seva

popularitat en les cerques fetes a Internet.

- Struts
- Spring MVC
- JSF (Reference Implementation de Sun Microsystems)
- RichFaces (JBoss Community)
- ZK
- GWT
- Grails
- Play!

1. Apache Struts

Struts és un marc de treball per donar suport al desenvolupament d'aplicacions web baix el patró MVC baix la plataforma Java EE.

Struts es va desenvolupar com a part del projecte Jakarta de l'Apache Software Fundation. Actualment es un projecte independent conegut com Apache Struts.

Una de les característiques principals d'Struts es reduir el temps de desenvolupament de les aplicacions.

Apache Struts és un projecte de software lliure i manté la compatibilitat amb totes les plataformes en les que es poden executar aplicacions amb Java EE.

Struts disposa de dues versions, la 1 i la 2. La versió 1 va ser el primer marc de treball de Java EE amb MVC, és el més antic. Amb l'aparició de la versió 2, es van introduir millores sobre la primera versió, de cara a simplificar las accions més repetitives, així com la seva integració amb AJAX, etc. Aquesta segona versió, sobretot va proporcionar més competitivitat a aquest marc de treball en vers en nous que anaven sorgint.

Struts es basa en el patró de disseny del model-vista-controlador (MVC), el qual s'utilitza àmpliament, es considerat de gran solidesa i es considerada una solució estàndard en la construcció d'aplicacions web.

2. Spring MVC

Spring es un marc de treball de codi obert per desenvolupar aplicacions Java. La primera versió va ser escrita per Rod Jonhson, qui la va llançar primer amb la publicació del seu llibre Expert One-on-One Java EE Design and Development (Wrox Press, octubre 2002).

Spring sortir inicialment baix Apache 2.0 License en juny de 2003. El primer gran llançament fou la versió 1.0, que aparegué en març de 2004.

Encara que Spring Framework no obliga a emprar un model de programació en particular, s'ha popularitzat dins la comunitat de programadors Java al considerar una alternativa i substitut del model d'Enterprise JavaBean (EJB).

Per el seu disseny de marc de treball ofereix molta llibertat als desenvolupadors en Java

i solucions molt ben documentades i fàcils d'emprar per les pràctiques comuns dins l'indústria.

Per Spring existeixen moltes extensions i millores per construir aplicacions basades en web per damunt de la plataforma empresarial de Java (Java Enterprise Platform).

3. JSF (Reference Implementation de Sun Microsystems)

JavaServer Faces (JSF) és una tecnologia basada en web que simplifica el desenvolupament d'interfícies d'usuari en aplicacions Java EE. JSF empra JavaServer Pages (JSP) com la tecnologia que permet fer el desplegament de las pàgines, però també se poden acomodar a altres tecnologies com XUL.

Realment JSF és una especificació, es a dir, necessita una implementació per poder-se executar en els servidors Java EE. L'especificació de JSF fou desenvolupada per la Java Community Process com JSR 127 (JSF 1.0 i 1.1), JSR 252 (JSF 1.2) i JSR 314 (JSF 2.0).

JSF inclou:

- Un conjunt d'APIs per representar components de l'interfície d'usuari i administrar el seu estat, manejar events, validar entrada, definir un esquema de navegació de les pàgines i donar suport per internacionalització i accessibilitat.
- Un conjunt per defecte de components per a l'interfície d'usuari.
- Dues biblioteques d'etiquetes personalitzades per JavaServer Pages que permeten expressar un interfície JavaServer Faces dins d'una pàgina JSP.
- Un model d'events en el costat del servidor.
- Administració d'estats.
- Beans administratius.

Objectius de disseny:

- Definir un conjunt simple de classes base de Java per components de l'interfície d'usuari, estat dels components i events d'entrada. Aquestes classes tractaran els aspectes del cicle de vida de l'interfície d'usuari, controlant l'estat d'un component durant el cicle de vida de la seva pàgina.
- Proporcionar un conjunt de components per l'interfície d'usuari, incloent els elements estàndards de HTML per representar un formulari. Aquests components s'obtindran d'un conjunt bàsic de classes base que se poden utilitzar per definir components nous.
- Proporcionar un model de JavaBeans per enviar events des dels controls de l'interfície d'usuari del costat del client a l'aplicació del servidor.
- Definir APIs per la validació d'entrada, incloent suport per la validació en el costat del client.
- Especificar un model per l'internacionalització i localització de l'interfície d'usuari.
- Automatitzar la generació de sortides apropiades per l'objectiu del client, tenint en conte totes les dades de configuració disponibles del client, com versió del navegador, etc.

4. RichFaces (JBoss Community)

RichFaces és una biblioteca open source de components Ajax per JavaServer Faces, organitzada per JBoss.org. Permet una fàcil integració de les capacitats d'Ajax en el desenvolupament d'aplicacions empresarials Java EE.

RichFaces és més que una biblioteca de components de JavaServer Faces, ja que hi afegeix:

- Skinability (fàcilment el canvi i actualització de l'aplicació look and feel)
- Desenvolupament de Components de Kit (CDK) per ajudar en la construcció de components JavaServer Faces
- Marc de recursos dinàmics
- Tant la pàgina d'ample, i el component de control de components basats en Ajax.

RichFaces es va originar a partir d'un altra marc de treball anomenat Ajax4jsf al 2005. En el 2006 es va dividir Ajax4jsf amb el nou marc de treball RichFaces. Un any més tard, al 2007 es varen tornar ajuntar baix el nom de RichFaces.

El marc s'implementa com una biblioteca de components que afegeix la capacitat d'Ajax dins pàgines existents, de manera que un desenvolupador no ha d'escriure cap codi JavaScript o pot substituir components existents amb nous components Ajax. RichFaces permet la compatibilitat amb Ajax en pàgines en lloc del tradicional suport de components. Per tant, un desenvolupador pot definir l'esdeveniment en la pàgina que invoca una petició Ajax i les àrees de la pàgina que ha d'estar sincronitzat amb l'arbre de components JSF després dels canvis petició Ajax les dades en el servidor d'acord amb els esdeveniments disparats en el client.

RichFaces permet definir (per mitjà d'etiquetes JSF) diferents parts d'una pàgina JSF que voleu actualitzar amb una petició Ajax, i ofereix algunes opcions per enviar peticions Ajax al servidor. També la pàgina JSF no canvia d'un JSF de la pàgina i no cal escriure cap codi JavaScript a mà. Com que el control es del costat del servidor, no es necessari escriure JavaScript i l'estat de la pàgina es pot mantenir fàcilment al servidor.

5. ZK

ZK és un marc de treball per aplicacions web en AJAX, completament en Java, de programari de codi obert, que permet una completa interfície d'usuari per a aplicacions web sense utilitzar JavaScript i amb poca programació.

El nucli de ZK és un mecanisme conduït per esdeveniments basat en AJAX, sustentat sobre 70 components XUL i 80 components XHTML, i un llenguatge de marcatge per a dissenyar interfícies d'usuari. Els programadors dissenyen les pàgines de la seva aplicació en components XUL / XHTML rics en característiques, i els manipulen amb esdeveniments disparats per l'activitat de l'usuari final. És similar al model de programació trobat en les aplicacions basades en GUI d'escriptori.

ZK utilitza l'acostament anomenat centrat-en-el-servidor per a la sincronització de components i el pipelining entre clients i servidors es faci automàticament pel motor, i els codis d'Ajax siguin completament transparents per als desenvolupadors d'aplicacions

web. Per tant, els usuaris finals obtenen una interacció i resposta semblant a les d'una aplicació d'escriptori, mentre que la complexitat del desenvolupament és similar a la que tindria la codificació d'aplicacions d'escriptori.

A més de la programació basada en components i orientació a esdeveniments, de manera similar a Swing, ZK suporta un llenguatge de marcatge per a la definició d'una potent interfície d'usuari anomenada ZUML.

- ZUML està dissenyat perquè desenvolupadors no experts dissenyin interfícies d'usuari de forma eficient.
- ZUML permet a un desenvolupador barrejar diferents tipus de llenguatge de marcatge, com ara el llenguatge XUL de Mozilla i XHTML, tots ells en la mateixa pàgina.
- ZUML permet als desenvolupadors incloure scripts en llenguatge Java (interpretat per BeanShell) i utilitzar expressions EL per manipular els components i accedir a les dades.

6. GWT

GWT o Google Web Toolkit és un marc de treball creat per Google que permet amagar la complexitat de diversos aspectes de la tecnologia AJAX. És compatible amb diversos navegadors, i això és notori ja que cada navegador sol necessitar codi específic per aconseguir un front-end correcte en una aplicació web.

El concepte de Google Web Toolkit és bastant senzill, bàsicament el que s'ha de fer és crear el codi en Java usant qualsevol entorn de desenvolupament (IDE) de Java i el compilador ho traduirà a HTML i JavaScript.

Quan una aplicació és desplegada, el compilador GWT tradueix l'aplicació Java a un arxiu JavaScript, que pot ser ofuscat per optimitzar el rendiment.

GWT no és només una interfície de programació; proporciona un conjunt d'eines que permeten desenvolupar funcionalitats JavaScript d'alt rendiment en el navegador del client.

Una aplicació GWT pot ser executada en dues maneres:

- Mode host (Hosted mode): L'aplicació s'executa com a codi bytecode de Java dins de la màquina virtual de Java (JVM). Aquesta manera és el més usat per a desenvolupament, suportant el canvi de codi en calent i la depuració.
- Mode web (Web mode): L'aplicació s'executa com a codi JavaScript i HTML pur, compilat a partir del codi Java. Aquesta manera se sol utilitzar per al desplegament de l'aplicació.

La utilitat de línia d'ordres applicationCreator genera automàticament tots els fitxers necessaris per iniciar un projecte GWT, fins i tot permet crear un projecte per a Eclipse.

Hi ha diversos plugins de codi obert per ajudar a desenvolupar en diferents entorns de desenvolupament, com GWT4NB per NetBeans, Cypal Studio for GWT per Eclipse o gwtDeveloper per JDeveloper.

7. Grails

Grails és un marc de treball lliure per a aplicacions web desenvolupat sobre el llenguatge de programació Groovy (el qual es basa en la Java). Grails pretén ser un marc de treball altament productiu seguint paradigmes com ara convenció sobre configuració o no et repeteixis (DRY), proporcionant un entorn de desenvolupament estandarditzat i ocultant gran part dels detalls de configuració al programador.

Grails ha estat impulsat principalment per l'empresa G2One, la qual va ser adquirida per la desenvolupadora de programari lliure SpringSource el novembre de 2008. A l'agost de 2009 SpringSource va ser al seu torn adquirida per VMware, empresa especialitzada en virtualització de sistemes.

Grails va ser conegut com 'Groovy on Rails' (el nom va canviar en resposta a la comanda de David Heinemeier Hansson, fundador de Ruby on Rails). Es va iniciar el juliol de 2005, amb la versió 0.1 29 març 2006 i la versió 1.0 anunciada el 18 de febrer de 2008. El desembre de 2009 es va publicar la versió 1.2, i el maig de 2010 la versió 1.3.

Grails s'ha desenvolupat amb una sèrie d'objectius en ment:

- Oferir un marc de treball web d'alta productivitat per a la plataforma Java.
- Reutilitzar tecnologies Java ja provades com Hibernate i Spring sota una interfície simple i consistent.
- Oferir un marc de treball consistent que redueixi la confusió i que sigui fàcil d'aprendre.
- Oferir documentació per a les parts del marc de treball rellevants per als seus usuaris.
- Proporcionar el que els usuaris necessiten en àrees que sovint són complexes i inconsistents:
    - Framework de persistència potent i consistent.
    - Patrons de visualització potents i fàcils d'usar amb GSP (Groovy Server Pages).
    - Biblioteques d'etiquetes dinàmiques per crear fàcilment components web.
    - Bon suport d'Ajax que sigui fàcil d'estendre i personalitzar.
- Proporcionar aplicacions exemple que mostren la potència del marc de treball.
- Proporcionar un entorn de desenvolupament orientat a proves.
- Proporciona una entorn complet de desenvolupament, incloent un servidor web i recàrrega automàtica de recursos.

Grails s'ha dissenyat per ser fàcil d'aprendre, fàcil per desenvolupar aplicacions i extensible. Intenta oferir el balanç adequat entre consistència i funcionalitats potents.

Alta productivitat

Grails té tres característiques que intenten incrementar la seva productivitat comparant-lo amb els marc de treball Java tradicionals:

- Inexistència de configuració XML.
- Entorn de desenvolupament preparat per a funcionar des del primer moment.
- Funcionalitat disponible mitjançant mètodes dinàmics.

8. Play!

Play! és un marc de treball web fet per a Java que ens permet fer moltes coses seguint els paradigmes d'altres marcs de treball de desenvolupament web àgil, entre els quals entren les idees d'altres llenguatges com Python, Ruby o Groovy.

A més, és un marc de treball Java, el que moltes vegades ens frena d'usar certes tecnologies de desenvolupament àgil (com Grails o Pylons) és el llenguatge de programació. Sens dubte al web Java és el més utilitzat a nivell mundial en empreses grans, per tant un marc de treball de Python, de Ruby o de qualsevol altre llenguatge que no sigui Java, no és una opció.

És Java estàndard, el que permet que puguem utilitzar qualsevol llibreria Java. Així com és possible combinar certs marcs de treball i tecnologies conegudes com WebFlow, Maven, ant, Ivy i molts altres, per als quals hi ha un mòdul.

No és Java EE, el que no és un desavantatge, ja que el seu desplegament pot fer-se com qualsevol altra aplicació EE (en un war pujat al webcontainer). No és EE perquè no compleix amb l'especificació. No obstant això ajuda a que sigui el que Play! és: un marc de treball de màxima productivitat.

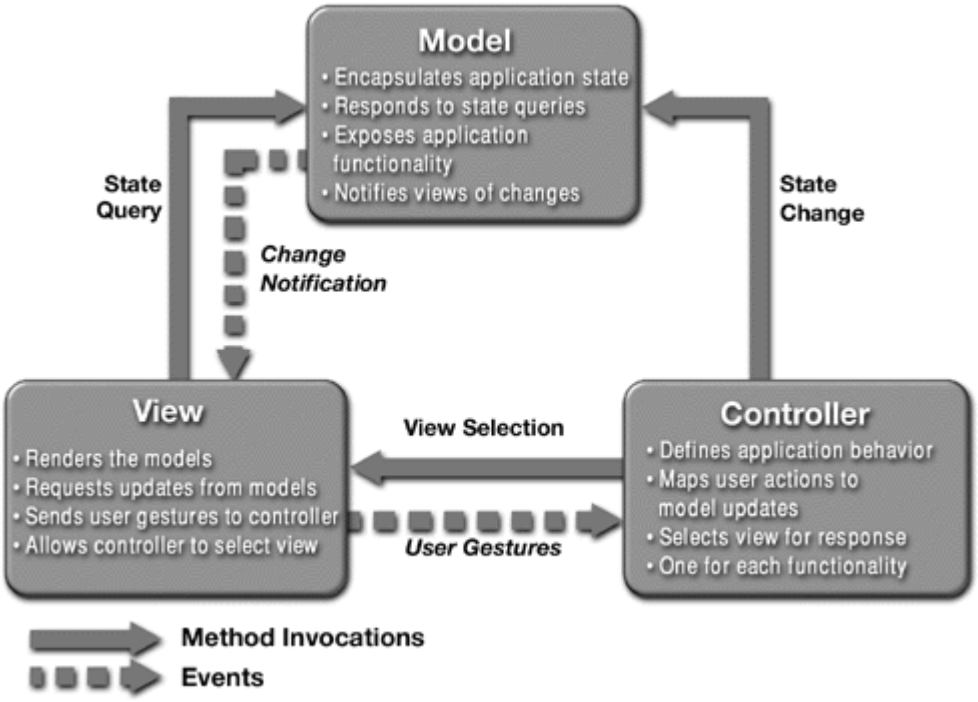
¿Errors?, corregeix i recàrrega el navegador. Potser aquest (en la meva opinió) és una de les millors prestacions de Play!, ja que ens permet veure errors d'una manera molt explícita:

I només ens bastaria amb anar a corregir-lo, i recarregant el navegador ens és possible veure els canvis.

1.2.2 Patró MVC

1.2.2.1 Explicació del patró MVC

El patró de disseny model-vista-controlador (MVC) és la solució més implementada per la capa de presentació, ja que proporciona moltes avantatges a l'hora de construir i mantenir aplicacions software basades en web.



El model-vista-controlador es considera un patró d'arquitectura de software i es pot considerar una solució estàndard.

El patró MVC fou descrit per Trygve Reenskaug quan treballa amb el llenguatge Smalltalk en els laboratoris de Xerox. Des de principis d'aquest segle va anar guanyant seguidors fins a consolidar-se com la referència per els marc de treball desenvolupats per aplicacions web.

Aquest patró separa les dades de l'aplicació, l'interfície d'usuari i la lògica de control; es a dir, entre el model de dades persistit dins el servidor, la vista a visualitzar al navegador de l'usuari i les controladores per gestionar el tràfic de les peticions dels usuaris. Aquest fet va ajudar a construir aplicacions amb un nivell més baix d'acoblament, a identificar més correctament l'estructura de l'aplicació, a tenir codi més net i fàcil de construir i mantenir, a facilitar el treball en equip ja que cada perfil es pot dedicar a la part que coneix més, a utilitzar la millor solució de les existents per cada una de les parts i canviar-la fàcilment en cas de sorgir la necessitat.

Com a puntualització, la part que es considera el model dins MVC es correspon amb el model d'entitats de l'aplicació, la lògica de negoci i la persistència d'aquestes dades dins una base de dades típicament relacional.

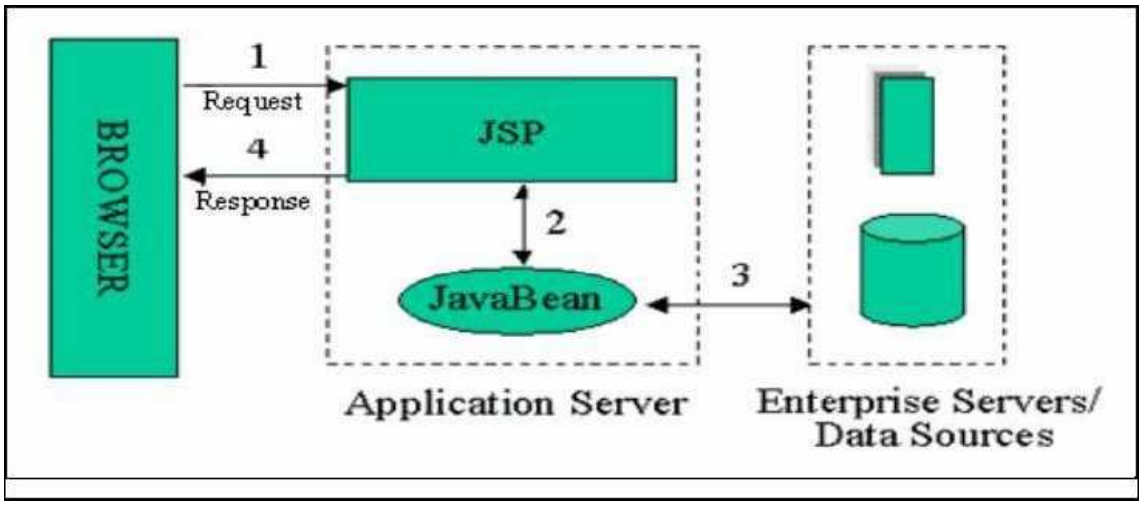
En definitiva, els marcs de treballs basats amb el MVC no fan més que reforçar totes aquestes característiques, a més de proporcionar una solució fàcilment identificable per els desenvolupadors.

1.2.2.2 Tipus de MVC

Hi ha dos tipus d'implementació del MVC, l'anomenat tipus I, i el tipus II.

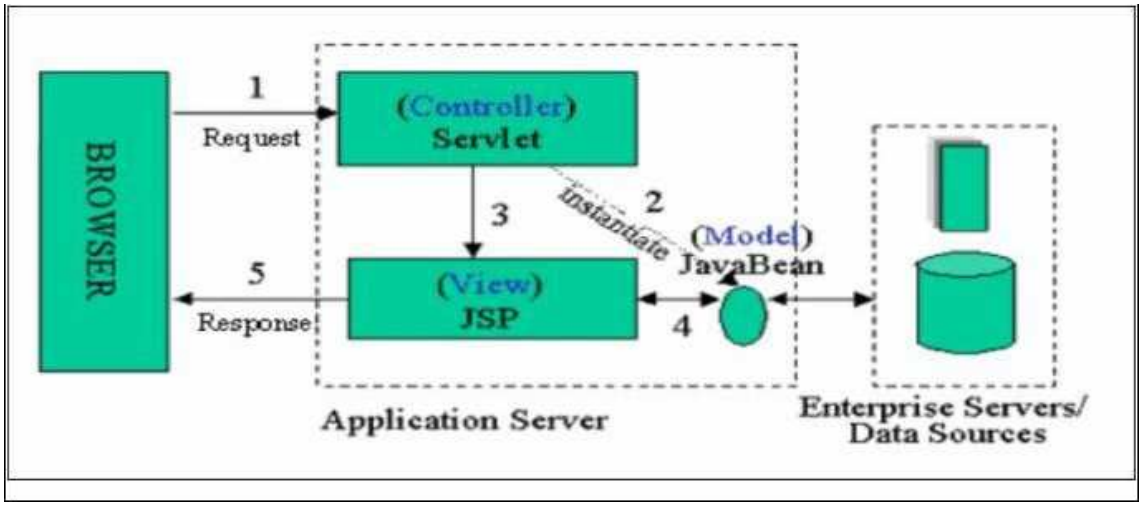
Tipus I

El tipus I va ser el primer en aparèixer. Implementa el patró MVC amb les JSP (Java Server Pages) que varen sorgir per resoldre les restriccions del CGI. En un primer moment es va abusar de les JSP i no hi havia una distinció clara entre la vista i el controlador. Més tard es varen separen aquest dos conceptes en dues JSP diferents millorant aquest aspecte. Es coneix aquesta solució com arquitectura centralitzada en la pàgina.



El tipus II

El tipus II, va sorgir per la necessitat de separar totalment la vista de la controladora i proporcionar una solució clara i simple. En aquest model s'introdueix la tecnologia dels servlets que són classes Java capaces de rebre peticions HTTP del navegador i gestionar-les segons el mètode d'invocació, ja sigui per GET o per POST. Aquestes noves classes seran les controladores, que rebran sol·licituds dels usuaris, recuperaran les dades del model, executaran la lògica de negoci programada i retornaran una resposta a la vista de la capa de presentació, en aquest cas les JSP que ara només contindran la vista. També es coneix aquesta solució com arquitectura basada



1.2.2.3 Relacionar el MVC amb la solució aportada per els marcs de treball

No tots els marcs de treball han implementat el MVC com a solució per separar la vista del negoci en aplicacions web.

De tots els marcs de treball esmentats, tenim una agrupació que sí apliquen el MVC, sobretot el tipus II. També tenim l'agrupació de marcs de treball que implementen JSF per la capa de presentació per aportar a les interfícies web més potencia, aquests marcs de treball empren el MVC de tipus I.

Els marcs de treball que implementen el MVC proporcionen als desenvolupadors totes les classes i fitxers de configuració necessaris per donar d'alta les converses entre el client i el servidor.

Per exemple, Struts proporciona el servlet ActionServlet que centralitza la conversa entre el client i el servidor. Aquest servlet només s'ha de donar d'alta dins el fitxer del projecte web.xml. D'aquesta manera el servlet capturarà totes les peticions i les traduirà per realitzar les següents accions. Aquesta informació està introduïda per el desenvolupador dins el fitxer anomenat struts-config.xml, on s'inclou el mapeig entre el path de la URL i l'acció a dur a terme l'aplicació. A més, dins aquest fitxer també es relaciona cada acció amb dues classes més, una per mantenir en sessió el formulari web i l'altra per agrupar les possibles accions a realitzar per l'usuari amb el formulari anterior. Finalment, també s'associa la resposta a l'usuari amb diferents vistes.

1.2.3 Valoracions

1.2.3.1 Característiques dels marcs de treball

Per començar, tenim les avantatges pròpies dels marcs de treball, són característiques que no es podem mesurar amb el codi sinó una conseqüència visible en el resultats.

Des d'un punt de vista econòmic un marc de treball ens ha de proporcionar

desenvolupaments més barats. Tota empresa vol aconseguir beneficis i construir aplicacions menys cares que donen més marge de benefici. Aquest fet també poc ajudar als projectes a tenir més marge de reacció a les possibles incidències o canvis dins el projecte.

Una altre conseqüència d'emprar un marc de treball es aconseguir els mateixos resultats amb menys temps. Aquesta característica està fortament relacionada amb el cost, ja que a menys temps, menys cost. Desenvolupar amb un marc de treball ens ha d'assegurar aquesta avantatge.

Per aconseguir disminuir el temps de producció, els marcs de treball proporcionen eines per automatitzar processos, reutilització de codi, minimitzacions d'accions repetitives, configuracions ja establertes, tutorials per optimitzar el desenvolupament, etc. En conclusió, un marc de treball proporciona valor afegit amb un punt de partida més avançat que si es comences a programar de cero.

Un dels problemes existents en tota aplicació és l'aparició d'incidència un cop l'aplicació s'ha posat en producció. Un marc de treball ha de disminuir el nombre de possibles incidències i el temps que es tarda amb la seva resolució. Un marc de treball sol proporcionar feina feta que ja està ben provada i evita les seves possibles incidències. A més, també solen proporcionar l'integració amb utilitats de testing com JUnit o Selenium per realitzar i validar les proves unitàries.

Una altre avantatge dels marcs de treball per el desenvolupament i manteniment de les aplicacions és l'implementació d'especificacions, utilització d'estàndards, ús correcta de la nomenclatura, restriccions de segons quines accions, encaminament d'algunes solucions d'una única forma, etc. D'aquesta manera, els diferents desenvolupadors del projecte no crearan solucions ja existents, no duplicaran la mateixa solució de forma diferent, podran incorporar-se a altres projectes amb el mateix marc de treball

L'utilització del mateix marc de treball dins diferents projectes minimitza l'adaptació d'un recurs ja format d'una aplicació a un altre. Aquesta característica és molt atractiva per els perfils més alts, ja que per els gerents i directius disminueix el temps de formació i conseqüentment el cost, i per els caps de projecte poden moure recursos entre varis projectes sense que això suposi una problemàtica especial.

Cal recalcar, que un marc de treball proporciona codi que no s'ha de tornar a programar, ja sigui en forma de plantilles, extensió o implementació de les classes proporcionades, configuracions ja realitzades, catàlegs de components, cross-browsing, internacionalització, etc. Així doncs, la posta en marxa d'un projecte software ha de ser més ràpida que si la comencessin de cero.

1.2.3.2 Altres característiques

Internacionalització

La internacionalització o i18n és una de les característica bàsiques a les quals ha de donar suport un marc de treball. La majoria d'aplicacions web són multi-idioma, tant les comercials com les privades. És molt més pràctic integrar l'internacionalització com

una característica més del marc de treball i evitar al desenvolupador haver de preocupar-se d'aquesta funcionalitat.

Geolocalització

Una altra característica es la geolocalització, conèixer des d'on accedeix un usuari a l'aplicació. Integrar-ho dins el marc de treball simplifica al desenvolupar l'accés d'aquesta informació.

Càrrega de dades

Els marc de treball permeten simplificar la comunicació dels formularis HTML amb la càrrega de les dades dins els beans del model de dades.

Aquesta conversió tan farragosa s'hauria de fer cada vegada que s'envia una petició al servidor i aquest retorna la resposta. Al llançar la petició, s'envia el formulari dins la petició HTTP. Una vegada al servidor, aquest ha de recuperar el formulari i cada un dels camps enviats que necessita per realitzar l'acció pertinent. A la vista, també s'ha d'indicar a cada in put el valor del servidor perquè sinó es perdria la persistència de les dades.

Els marcs de treball permeten de diferents maneres que aquestes conversions siguin automàtiques i els desenvolupadors no s'hagin de preocupar d'aquestes conversions i persistència de les dades en sessió. Alguns empren uns tags especials que es compilen en la part del servidor i mantenen aquesta comunicació. D'altres proporcionen uns atributs especials als tags HTML per indicar al servidor com fer aquesta conversió.

Plantilles

Una altre característica que proporcionen els marcs de treball es la construcció de les vistes en forma de plantilles.

Totes les pàgines web, tant si son portals, tendes, comercials o back-office, solen tenir una estructura repetida al llarg de la web. Així doncs, podem definir les pàgines com a plantilles i incloure a cada una de les seves parts un component reutilitzable. Aquesta pràctica és molt útil ja que ens permet reutilitzar codi, dotar a cada component totes les característiques i events per fer-los autònoms, conseguir un disseny net, clar, simple, fàcil de modificar i mantenir.

Validacions

Els marcs de treball web tenen molta interacció amb formularis amb dades que s'han de validar. Normalment, el programador té dues opcions: fer-ho per JavaScript abans de submitar el formulari, o fer-ho per java una vegada s'han recuperat les dades del request.

En aquest cas el marcs de treball proporcionen alguna forma de configuració on s'indica el formulari, el camp i el tipus de valor o validació. Així doncs és molt més fàcil validar un formulari ja que no s'ha de programar una vegada rera l'altra el mateix codi. Habitualment, es proporcionen validacions bàsiques com camps de text, numèrics, dnis,

telèfons, etc. Però també podem ampliar-ho amb les nostres pròpies validacions.

Notificacions

En qualsevol aplicació, ja sigui web o no, qualsevol acció de l'usuari espera una resposta, ja sigui carregar una pàgina, insertar un valor o consultar una taula. Per informar a l'usuari de l'èxit o el fracàs de la seva petició utilitzen notificacions que es mostren a la vista, o pàgina web carregada al navegador.

Els marcs de treball proporcionen els mecanismes necessaris per assignar aquests events a la resposta i altres per visualitzar-los per pantalla.

Aquestes eines simplifiquen la feina dels desenvolupadors, disminueixen el temps de programació i proporcionen una forma estàndard d'entrada/sortida de missatges entre l'usuari i el servidor.

1.2.3.3 Llistat de les característiques més importants

Una vegada hem presentat els diferents marcs de treball presents a Internet, els hem agrupat a partir d'algun criteri comú i analitzat alguns d'ells amb una mica més de profunditat.

Durant aquest procés hem pogut relacionar ràpidament punts en comú que llistarem a continuació com algunes de les característiques més importants que comparteixen els marcs treball.

- Disminució del cost del projecte
- Disminució del temps de desenvolupament
- Proporciona valor afegit de funcionalitats ja implementades.
- Disminució de les incidències a l'entorn de producció.
- Proporciona eines per testejar l'aplicació.
- Ús d'estàndards i tecnologies ja consolidades.
- Facilitat d'integració de nous recursos al projecte.
- Proporciona codi ja desenvolupat.
- Internacionalització.
- Geolocalització.
- Gestió automàtica de l'intercanvi d'informació entre client i servidor.
- Construir les pàgines webs com plantilles
- Validacions automàtiques de les dades del formular
- Notificacions dels errors i accions a la capa web

1.2.3.4 Relacionar les característiques amb els problemes que solucionen

Moltes de les característiques llistades en el punt anterior són requeriments bàsics de tota aplicació web, com per exemple l'internacionalització.

Moltes aplicacions web són portals comercials que han de tenir l'opció de visualitzar el seu contingut amb l'idioma de l'usuari que accedeix a la web.

Aquestes característiques es poden programar amb les eines que ens proporciona Java EE, però no deixen de ser un procés repetitiu on molta part del codi s'ha de replicar i pot provocar tenir codis diferents que fan la mateixa funcionalitat. Aquest aspecte és nefast per el manteniment posterior, l'evolució del projecte i l'escalabilitat de l'aplicació.

Emprar un marc de treball afavoreix a evitar aquests problemes, només existeix un codi, s'utilitza d'una forma determinada, segueix els estàndards i les metodologies del projecte i aplicació, és fàcilment escalable i mantenible.

1.2.3.5 Relacionar aquests problemes amb patrons de disseny

Cada una d'aquestes característiques resolen problemes concrets. Cada un d'aquests problemes són recurrents i apareixen a la majoria dels projectes.

Un dels motius de la creació dels patrons de disseny va ser precisament reutilitzar el coneixement aplicant solucions ja conegudes, correctes i validades per l'experiència en les etapes d'anàlisis i construcció dels projectes.

Aquests patrons de disseny facilitaven l'especificació o implementació d'alguns d'aquests problemes agilitzant el desenvolupant, estandaritzant les solucions i disminuint els riscs d'incidències posteriors.

A semblant-se dels patrons de disseny tenim els marcs de treball que també volen proporcionar als desenvolupadors solucions més ràpides i robustes sobre tasques repetitives, recurrents o farragoses per disminuir el temps de producció i d'incidències, així com millorar la qualitat del producte i homogeneïtzar l'estructura de les aplicacions i el codi desenvolupat.

1.3 Etapa d'especificació i implementació del marc de treball

1.3.1 Anàlisis del nou marc de treball

1.3.1.1 Abast del marc de treball

Primer de tot li hem de donar un nom per poder fer referència al marc de treball. S'ha decidit anomenar-lo LightWeightFramework, o LWF.

El nostre marc de treball serà molt senzill i només implementarà les característiques bàsiques que haurien de tenir tots els marcs de treballs per la capa de presentació.

Característiques que contindrà el nostre marc de treball LWF.

- Ús d'estàndards i tecnologies ja consolidades com java, jsp, xml, properties, servlet
- Centralitzar en una controladora les peticions HTTP.
- Gestió automàtica de l'intercanvi d'informació entre client i servidor.
- Validacions automàtiques de les dades del formulari.
- Construir les pàgines web com plantilles.
- Internacionalització dels texts.
- Notificacions dels errors i accions a la capa web.

El que no contindrà en nostre marc de treball LWF.

- Proporcionar valor afegit amb altres funcionalitats ja implementades.
- Proporcionar eines per testejar l'aplicació (ni junit, ni selenium, ni proves unitàries).
- Proporcionar codi ja desenvolupat (a part del mínim no proporciona res més).
- Geolocalització (per sabre on està l'usuari)
- Components custom per la presentació com en RIA o JSF (menú, tabulador, select, etc)

Altres característiques que complirà al complir les anteriors.

- Disminució del cost del projecte
- Disminució del temps de desenvolupament
- Disminució de les incidències a l'entorn de producció.
- Facilitat d'integració de nous recursos al projecte.

1.3.1.2 Comparació amb els marcs de treball actuals

Els marcs de treball actuals són molt complets i complexes. La majoria proporcionen moltíssimes funcionalitats i alguns d'ells es basen en altres mars de treballs més bàsics per poder oferir més característiques i més riques.

Un marc de treball sol tenir un o varis fitxers jar amb les llibreries necessàries per compilar i executar l'aplicació (a part de les del marc de treball pot necessitar altres jars).

També sol venir acompanyada d'una pàgina web oficial, de documentació del projecte del marc de treball, una guia de desenvolupament, d'altres tutorials, blocs i fòrums per els usuaris, i alguna característica més.

Cada marc de treball, o millor dit, cada versió d'un marc de treball està compilada per funcionar amb una versió de java (1.4, 5.0, 6.0) i per tant s'ha de triar la versió del servidor (Tomcat, JBoss) correcte, i tenir instal·lada la mateixa versió de la màquina virtual.

1.3.1.3 Llistat de requeriments funcionals/estructurals

Llistat de requeriments funcionals/estructurals.

- El marc de treball ha d'estar implementat amb tecnologies Java EE com java, jsp, xml, properties, servlets
- Les aplicacions que utilitzin el marc de treball han de poder executar-se damunt servidors amb contenidors J2EE.
- Ús d'estàndards i tecnologies ja consolidades com java, jsp, xml, properties, servlet
- Centralitzar en una controladora les peticions HTTP.
- Gestió automàtica de l'intercanvi d'informació entre client i servidor.
- Validacions automàtiques de les dades del formulari.
- Construir les pàgines web com plantilles.
- Internacionalització.
- Notificacions dels errors i accions a la capa web.

1.3.1.4 Llistat de casos d'ús (suportats per el marc de treball)

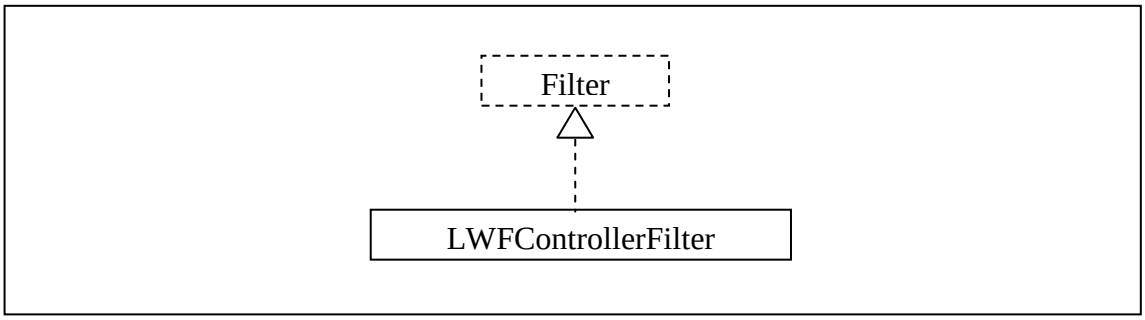
- Recuperar una petició HTTP de l'usuari i redireccionar-la a la lògica de negoci adequada.
- Poder crear les URLs amigables per l'usuari i per els motors de cerca com Google.
- Recuperació automàtica del formulari de la petició HTTP a la part servidor.
- Validació del format dels seus camps.
- Visualitzar la pàgina web amb l'idioma per defecte o el seleccionat per l'usuari.
- Visualitzar els errors a la capa web, també els missatges a l'usuari.

1.3.2 Disseny del nou framework

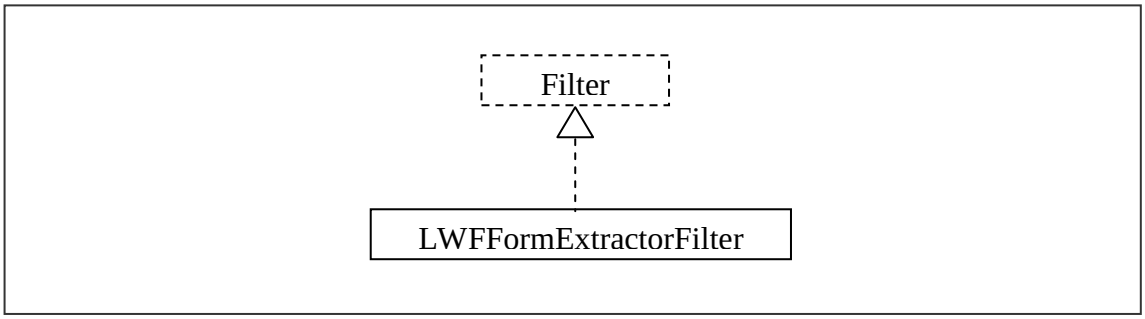
1.3.2.1 Diagrama de classes (UML)

En aquest apartat representarem els diagrames de classes que intervenen en el LWF.

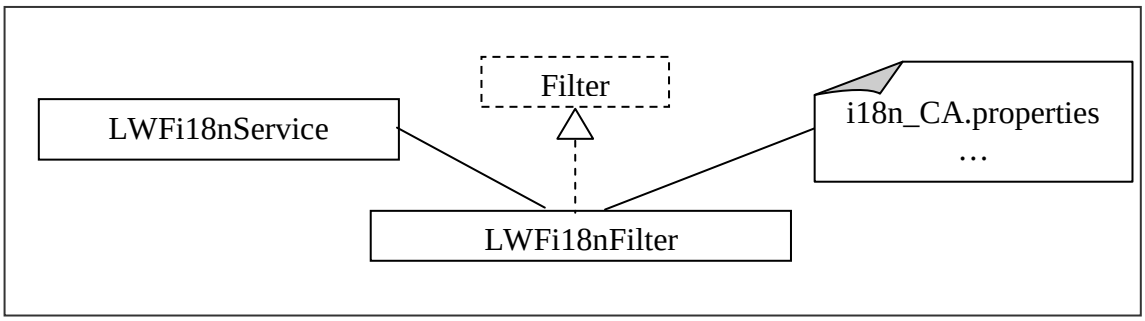
Controller



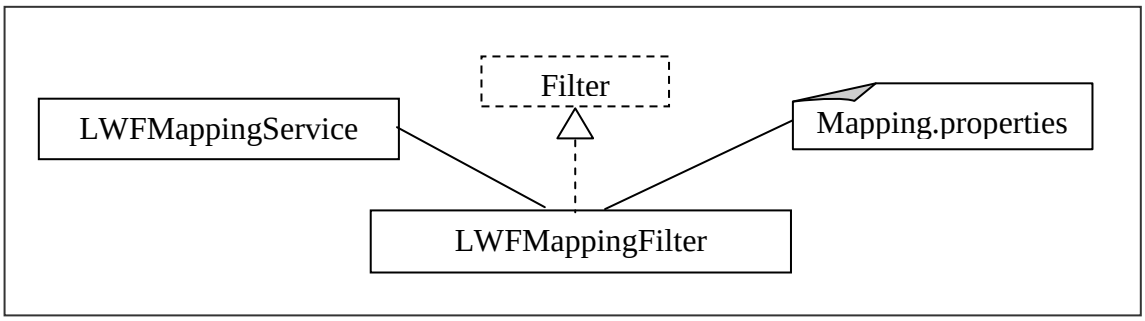
FormExtractor



i18n

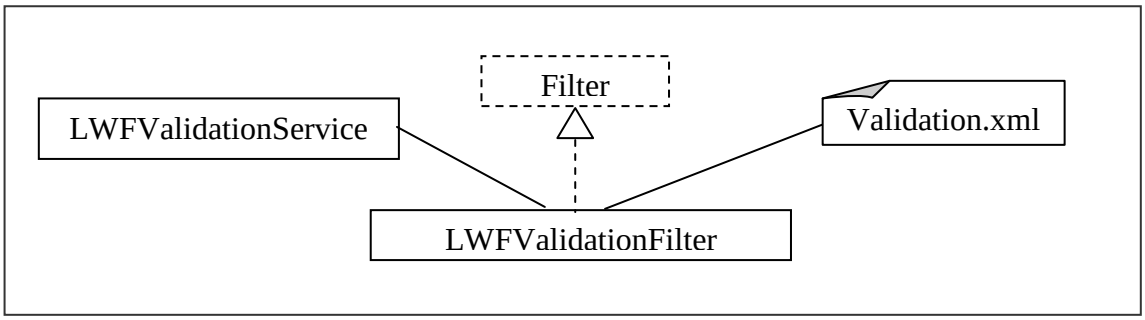


Mapping

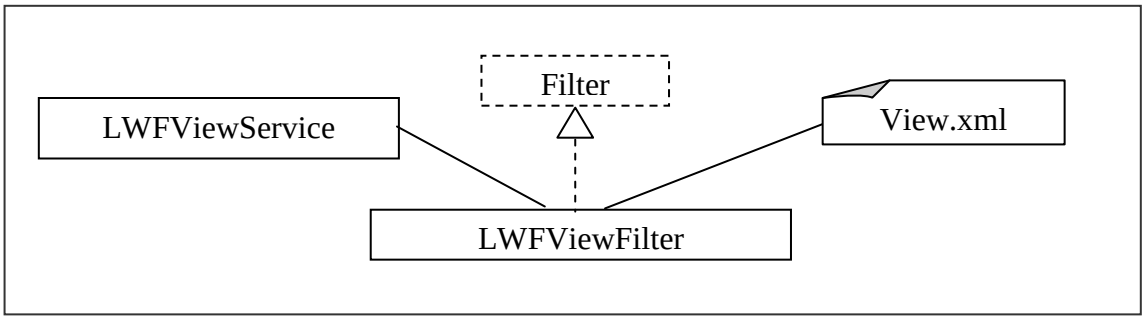


Message

Validation



View



1.3.2.2 Patrons de disseny

Els patrons de disseny emprats en el marc de treball LWF són els següents.

Intercepter filter

Per poder interceptar les peticions de l'usuari s'ha emprat aquest patró amb classes que implementen l'interface Filter. Mitjançant el mètode doFilter() podem interceptar cada petició d'usuari, fer els tractaments que necessitem i tornar-lo a enviar a la cadena d'esdeveniments.

Dins LWF tenim 4 classes que implementen el Filter:

- LWFControllerFilter.java
- LWFFormExtractorFilter.java
- LWFValidationFilter.java
- LWFMappingFilter.java

Front Controller

Com a part del MVC tenim una controladora, aquest controladora es pot considerar un patró individual anomenat Front Controller. Aquest patró, i en concret la classe LWFControllerFilter.java, centralitzen totes les peticions de l'usuari i les gestionen cap els següents punts dels esdeveniments.

MVC

Com la majoria de marcs de treball de la capa de presentació, s'ha implementat un model-vista-controlador. Aquest patró independitza la gestió de les peticions d'usuari, amb les dades i el negoci de l'aplicació i finalment amb la vista que es retornarà a l'usuari amb el resultat de la petició.

Com ja hem comentat abans, tenim la controladora implementada a la classe LWFControllerFilter.java i el mapeig a la classe que implementa el UserRequest. Les classes del model estaran ben separades i les vistes les formaran les plantilles de la vista.

Singleton

També podem considerar dues classes com l'implementació del patró singleton.

Aquestes son les següents.

- LWFValidationService.java
- LWFMappingService.java

Aquestes dues classes aporten serveis amb una instància única de l'objecte de la classe, encapsulen les dades referents a la classe Filter que aporten servei i només existeix una forma d'accés a les mateixes.

Factory reflection

Hi ha varies classes que utilitzen reflection per declarar e instanciar objectes de forma dinàmica en temps d'execució.

Podem considerar els mètodes que implementen aquest procés com a implementacions del patró Factory en l'àmbit del reflection.

Aquestes classes son:

- LWFFormExtractorFilter.java, recupera la classe java que representa el formulari i li seteja les dades enviades en els paràmetres del formulari HTTP.
- LWFValidationFilter.java, recupera el formulari java i mitjançant gets del camps intenta validar-los segons la configuració del fitxer Validation.xml i els mètodes de la classe LWFValidationMethods.java.
- LWFMappingService.java, recupera el path de la petició, el mapeja a una classe java mitjançant la configuració del fitxer Mapping.properties, l'invocació es fa al mètode UserRequest que han implementat les classes que gestionen les peticions dels usuaris i han implementat l'interface UserRequestInterface.java.

1.3.2.3 Aplicació del MVC

Per aplicar el MVC s'han d'assignar els rols de cada una de les tres parts (model, vista, controlador) a una funcionalitat representada al codi.

Per una banda tenim la part **controladora** que estarà representada per:

- LWFControllerFilter.java: rebrà les peticions HTTP de forma centralitzada per gestionar l'entrada/sortida entre els clients i el servidor. Per cada URL cercarà el mètode que s'ha d'executar.
- LWFMappingFilter.java: aquí es mapejarà el path de URL amb el mètode a executar d'una classe concreta.
- Mapping.properties: on es configurarà el mapeig del path amb les classes java.

Així doncs, els usuaris enviaran peticions HTTP al servidor, el primer que s'executarà serà el filter LWFControllerFilter.java que recuperarà el path de la URL i més tard el LWFMappingFilter.java mapejarà amb les dades configurades al propertis, d'aquí recuperarà una classe i un mètode que executarà esperant una resposta, una redirecció o un forward.

Per un altra banda tenim el **model**, on s'executa la lògica de negoci de l'aplicació damunt el model de dades que persistirem en una base de dades. Per aquesta part, el nostre marc de treball LWF no donarà suport.

L'únic que aportarà el LWF és una estandardització amb els noms emprats al model i el consell de no crear tota la lògica dins els mètodes invocats al mapeig, millor crear classes de negoci per aquesta funció. Així doncs podem definir el següent.

- *UserRequet.java on s'inclouran els mètodes del mapeig.
- *Bussines.java on s'inclouran les funcionalitats o requeriments que executarà l'aplicació.
- *Persist.java que gestionarà l'intercanvi de dades entre l'aplicació i la BBDD.
- *Entity.java classe tipus bean on es guarden les dades d'una entitat i es mapeja amb una taula d'una base de dades relacional. Cada atribut, normalment, serà una columna dins BBDD.

Finalment tenim la **vista,** que estarà formada per el següent.

- Un nom del tipus pantalla.manteniment.usuari on s'especificarà una jsp tipus plantilla i les jsp que la conformen, del tipus
    - View.xml
        - Pàgina=pantalla.manteniment.usuari
            - Plantilla=pantalla_manteniment.jsp
                - Element.cap=capçalera.jsp
                - Element.ms=menusuperior.jsp
                - Element.me=menuesquerra.jsp
                - Element.cos=manteniment_usuari.jsp
                - Element.peu=peu.jsp

Gràcies a aquesta estructura podem crear un pàgina com una plantilla. Cada una de les parts es pot reutilitzar en altres pàgines. La plantilla també pot ser genèrica i només canviar, per exemple el cos. L'ideal seria que cada pàgina pogués heretar d'una altre, així no faria falta duplicar la declaració dels elements repetits en cada pàgina.

Amb aquest punts ben senzills ja tenim implementat el MVC.

1.3.3 Implementació del nou framework

1.3.3.1 Crear l'estructura del framework

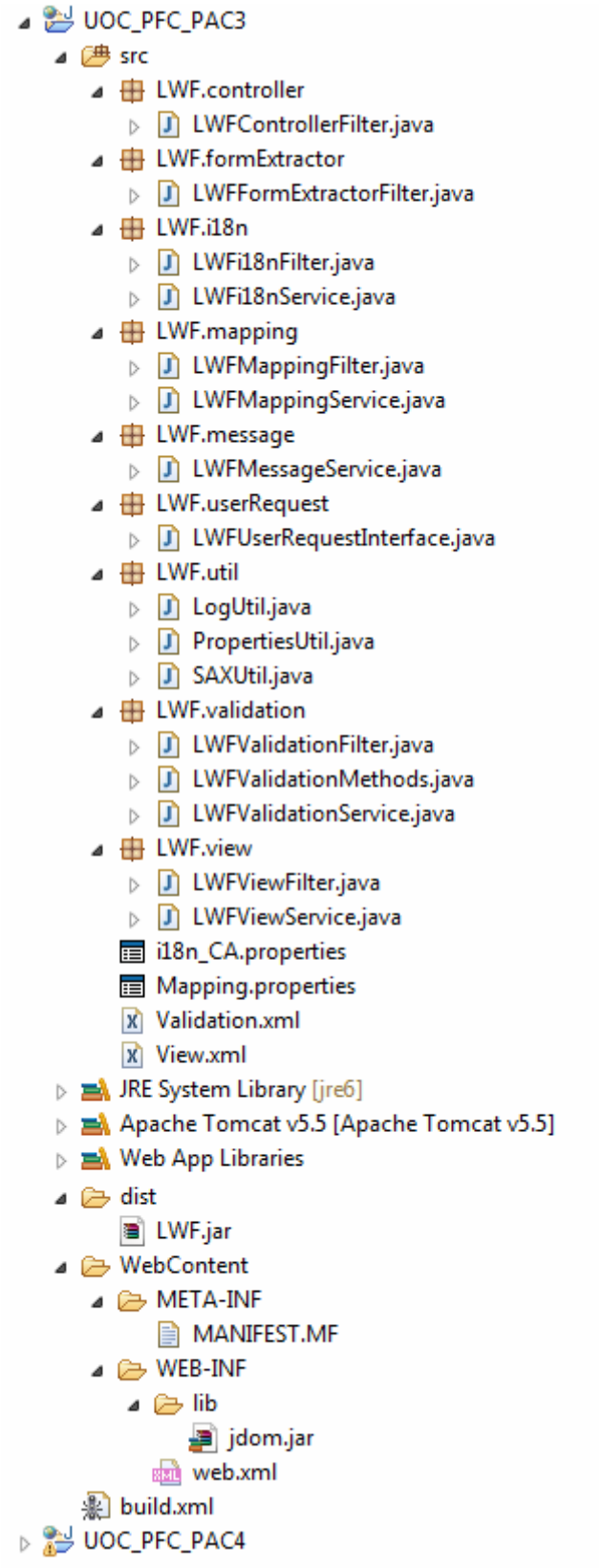
El marc de treball LWF es disposarà als desenvolupadors amb un fitxer jar anomenat **LWF.jar** que s'haurà d'incloure dins el projecte de la nova aplicació per poder compilar i executar; i també una guia d'ús a mode de tutorial, on s'explicarà com configurar l'aplicació per poder emprar el marc de treball.

Passos a seguir per emprar LWF.

- Incloure el LWF.jar dins la carpeta lib del nou projecte.
- Modificar el web.xml per incloure les clases del LWF.
- Crear els fitxers de configuració del LWF i anar actualitzant-los..
- Estendre o implementar les classes del LWF.

A continuació es presenta l'estructura del projecte dins Eclipse que ha servit per crear el marc de treball LWF.

- Aquest projecte s'ha creat com un Dynamic Web Project.
- Per poder compilar i executar el projecte s'ha configurat amb un Tomcat v5.5.
- A l'imatge es poden veure les classes que intervenen en el marc de treball.
- També podem veure els fitxers de configuració que haurem de reproduir a la nova aplicació.



1.3.3.2 Crear les classes interface i classes abstractes

Per poder crear un nou projecte a partir de LWF només necessitem implementar

l'interface UserRequestInterface.java a les classes que hagin d'executar les accions de l'usuari.

Aquest interface conté un mètode abstracte on s'ha d'implementar l'acció de l'usuari.

Així doncs quan un usuari fa la següent acció.

```
http://www.domini.com/contexte/accio-de-usuari
```

La classe de mapping llegeix d'un fitxer propertis quina classe es l'encarregada d'executar aquesta acció.

```
accio-de-usuari=com.aplicacio.AccioDeUsuariUserRequest
```

La classe AccioDeUsuariUserRequest implementa l'interface UserRequestInterface, on hi ha la classe abstracte UserRequest que invocarà el mapping.

```
package com.aplicacion.peticiones;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.aplicacion.formularios.LoginUserForm;

import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;

public class LoginUserRequest implements LWFUserRequestInterface
{
        @Override
        public void userRequest(HttpServletRequest req, HttpServletResponse resp)
        throws Exception
        {
               LogUtil.print("Login.userRequest()");
               LoginUserForm form = (LoginUserForm) req.getAttribute("LWF_FORM");
               LogUtil.print(form.getUser());
               LogUtil.print(form.getPassword());
        }
}
```

1.3.3.3 Crear els properties, xmls, per configurar el marc de treball

Al crear un nou projecte es necessari configurar-lo correctament, de tal manera que haurem de crear els fitxers de configuració que utilitza el marc de treball LWF per funcionar correctament.

Hem de crear els següents fitxers.

- Projecte\src\Mapping.properties
- Projecte\src\Validation.xml
- Projecte\src\View.xml
- Projecte\src\i18n_CA.properties

Al fitxer de mapping indicarem els path de les accions de l'usuari i les mapejarem amb les classes corresponents que l'implementaran.

```
login=com.aplicacio.LoginUserRequest
```

Al fitxer de validació indicarem quins camps del formulari volem validar i quines validacions s'han de realitzar.

```
<validation>
        <form name="LoginUserForm">
                <input name="user" method="isInteger"/>
                <input name="password" method="isInteger,isPasswordPattern"/>
        </form>
</validation>
```

Al fitxer de la vista construirem les pàgines emprades per l'aplicació en forma de plantilla. Existiran una seria de fitxers jsp que funcionaran com a plantilles i segons l'acció d'usuari s'inclouran una pàgines o dades diferents.

```
<view>
        <page name="pantalla.manteniment.usuari" template="pantalla_manteniment">
                <element name="cap" value="capçalera.jsp"/>
                <element name="ms" value="menusuperior.jsp"/>
                <element name="me" value="menuesquerra.jsp"/>
                <element name="cos" value="manteniment_usuari.jsp"/>
                <element name="peu" value="peu.jsp"/>
        </page>
</view>
```

Finalment tindrem fitxers properties per donar d'alta els textos emprats en les aplicacions.

```
missatge.login.error=La autentificació ha fallat! torna-ho a intentar...
missatge.login.ok=La autentificació ha sigut correcta!
missatge.error.validacio=Error en la validació
```

1.3.3.4 Implementar el marc de treball i deixar-lo llest per ser emprat

Per implementar el marc de treball LWF el més recomanable és emprar un IDE per ajudar-nos i facilitar-nos el desenvolupament, en el nostre cas s'ha decidit utilitzar Eclipse al ser gratuït i proporcionar-nos tota la funcionalitat requerida per desenvolupar el projecte.

Ens hem descarregat l'**Eclipse** des de la pàgina oficial, en concret la versió per JEE: **Eclipse IDE for Java EE Developers**.

http://www.eclipse.org/downloads/
http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2

Una vegada instal·lat Eclipse, hem de crear un projecte per desenvolupar LWF. De totes les possibilitats que ofereix Eclipse, hem seleccionat un **Dynamic Web Project** que ja

ens ofereix l'estructura correcta per una aplicació JEE, a més d'associar-lo a un servidor per poder executar l'aplicació mentrestant l'estem desenvolupant.

Per poder executar l'LWF també hem instal·lat el servidor Tomcat i un plugin a l'Eclipse per facilitar el desenvolupament.

El servidor **Tomcat**, amb la seva versió 5.5.25 l'hem descarregat de la web oficial d'Apache.

http://tomcat.apache.org/whichversion.html

El plugin d'Eclipse per Tomcat s'anomena **Sysdeo Tomcat Launcher Plugin 3.2.1** i ens l'hem descarregat de la pàgina oficial de plugins d'Eclipse.

http://marketplace.eclipse.org/content/sysdeo-tomcat-launcher-plugin

Una vegada tenim tot l'entorn muntat ja podem començar a desenvolupar les classes, interfaces i la resta de fitxers de configuració per LWF.

Al final del desenvolupament, i per poder emprar LWF a altres aplicacions, només hem de generar un fitxer jar que anomenarem **LWF.jar** que hauran d'importar com una llibreria més dins el projecte.

Projecte/WebContent/WEB-INF/lib/LWF.jar

A més d'aquest jar també s'haurà d'importar un altre que necessita LWF, es el **jdom.jar** que permet manipular xmls.

Projecte/WebContent/WEB-INF/lib/jdom.jar

Per acabar, només faltarà crear els fitxers de configuració dins la nova aplicació per el mapeig, validació i vista.

Projecte/src/Mapping.properties Projecte/src/Validation.xml Projecte/src/View.xml Projecte/src/i18n_CA.properties

1.3.3.5 Tutorial per emprar LWF per generar noves aplicacions

El marc de treball LWF aporta les següents funcionalitats:

Controller

LWF ens aporta una classe controladora per centralitzar les peticions HTTP dels usuaris de l'aplicació desenvolupada. Es a dir, totes les peticions que reb l'aplicació desenvolupada amb LWF passarien per aquesta classe. Aquesta classe estén de Filter,

que proporciona dos mètodes que s'executen al iniciar-se i aturar-se l'aplicació i un tercer que funciona com un listener, i es per on passen totes les peticions d'entrada i les seves respostes.

Aquesta classe ens ve dins el jar **LWF.jar**

```
LWF.controller.LWFControllerFilter.java
```

S'ha d'incloure dins el fitxer **web.xml** de l'aplicació a desenvolupar, amb l'ordre més alt

```
<filter>
        <filter-name>controller</filter-name>
        <filter-class>LWF.controller.LWFControllerFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>controller</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

A continuació presentem un exemple:

- Si tenim la URL: http:\\www.domini.com\contexte\pagina-mes-visitada
- Aquest filter recupera només el path, que en el nostra cas es pagina-mes-visitada, i després el puja al request per poder ser utilitzar per el mapper o alguna altra funcionalitat
- Recordem que una URL està formada per protocol+domini+port+contexte+path

FormExtractor

LWF proporciona una classe que, a partir del formulari que s'envia a la petició HTTP, seteja els seus valors dins una classe java. Perquè funcioni aquest procés s'ha d'enviar dins el formulari un camp amb el nom i package de la classe del formulari. Les classes dels formularis es recomana que s'anomenin amb el següent patró: *UserForm.java.

Aquesta classe ens ve dins el jar **LWF.jar**

```
LWF.formExtractor.LWFFormExtractorFilter.java
```

S'ha d'incloure dins el fitxer **web.xml** de l'aplicació a desenvolupar.

```
<filter>
        <filter-name>formExtractor</filter-name>
        <filter-class>LWF.formExtractor.LWFFormExtractorFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>formExtractor</filter-name>
        <url-pattern>/*</url-pattern>
```

```
</filter-mapping>
```

Perquè funcioni s'ha d'incloure el nom i package de la classe que inclou el **formulari**.

```
<input type="hidden" id="form" name="form"
        value="com.aplicacion.formularios.LoginUserForm"/>
```

Exemple de formulari html que s'enviarà a la petició http

```
<!-- El nom, id i method no son necessaris -->
<form action="http://localhost:8080/test/login">
        <input type="hidden" id="form" name="form"
                value="com.aplicacion.formularios.LoginUserForm"/><br/>
        <input type="text" id="user" name="user" value="AAAA"/><br/>
        <input type="text" id="password" name="password" value="1234"/><br/>
        <input type="submit"/>
</form>
```

Validation

LWF es proporciona una classe que valida els camps dels formulari java a partir d'un fitxer de configuració. També s'aporta una classe amb validacions per defecte que l'usuari pot estendre per emprar les seves pròpies validacions i configurar-les al fitxer xml.

Aquesta classe ens ve dins al jar **LWF.jar**

```
LWF.validation.LWFValidationFilter.java
```

També tenim una classe on s'emmagatzemen aquestes validacions proporcionant aquest servei al filter.

```
LWF.validation.LWFValidationService.java
```

L'altra classe que podem emprar o estendre es la següent.

```
LWF.validation.LWFValidationMethods.java
```

S'ha d'incloure dins el fitxer **web.xml** de l'aplicació a desenvolupar. Per poder validar hem d'emprar o que classe amb els mètodes de validació de LWF o una nostra.

```
<context-param>
        <param-name>validationMethodsClass</param-name>
        <param-value>LWF.validation.LWFValidationMethods</param-value>
</context-param>

<filter>
        <filter-name>validation</filter-name>
        <filter-class>LWF.validation.LWFValidationFilter</filter-class>
```

```
</filter>

<filter-mapping>
        <filter-name>validation</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Per poder realitzar la validació hem de crear els següents fitxers al projecte que volem desenvolupar.

```
/aplicació/src/Validation.xml
```

Exemple de **Validation.xml**

```
<validation>
        <form name="LoginUserForm">
                <input name="user" method="isMandatory,isString"/>
                <input name="password"
                        method="isMandatory,isString,isPasswordPattern"/>
        </form>
</validation>
```

Mapping

LWF també aporta l'implementació de URLs amigables, això significa que el path aplicat al navegador pot fer-se servir per ser més amigable o fàcil d'entendre per l'usuari. Aquesta classe recupera aquest path i el mapeja amb una classe java que executarà l'acció de l'usuari. Les classes que executaran les peticions dels usuaris es recomana que s'anomenin amb el següent patró: *UserRequest.java i han d'implementar l'interface UserRequestInterface, on es proporciona el mètode que invocarà el filter. Cada mètode d'aquestes classes serà una acció d'usuari o cas d'ús. El paràmetres d'entrada seran els mateixos que el mètode doPost/doGet dels servlets.

Aquesta classe ens ve dins al jar **LWF.jar**.

```
LWF.mapping.LWFMappingFilter.java
```

També tenim una classe on s'emmagatzema aquest mapeig proporcionant aquest servei al filter.

```
LWF.mapping.LWFMappingService.java
```

S'ha d'incloure dins el fitxer web.xml de l'aplicació a desenvolupar. Per poder validar hem d'emprar o que classe amb els mètodes de validació de LWF o una nostra.

```
<filter>
        <filter-name>mapping</filter-name>
        <filter-class>LWF.mapping.LWFMappingFilter</filter-class>
</filter>
```

```
<filter-mapping>
        <filter-name>mapping</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Aquest mapping s'enregistrarà creant un fitxer de tipus propertis al projecte que volem desenvolupar.

```
/aplicació/src/Mapping.properties
```

Exemple de Mapping.properties

```
login = com.aplicacion.peticiones.LoginUserRequest
autentificacion = com.aplicacion.peticiones.LoginUserRequest
home = com.aplicacion.peticiones.HomeUserRequest
```

Totes les classes *UserRequest han d'implementar l'interface UserRequestInterface. Cada una d'aquestes classes ha d'implementar el mètode abstracta userRequest que proporciona l'interface i que es invocat al mapejar-se el path per la classe LWFMappingFilter.java

Aquesta classe ens ve dins al jar **LWF.jar**.

```
LWF.userRequest.LWFUserRequestFilter.java
```

View

Finalment, també es proporcionarà la creació de pàgines web mitjançant plantilles. Amb el suport d'un fitxer de configuració es podran crear plantilles de pàgines web amb caixes buides, després crearem les pàgines web assignant un nom concret, relacionant-los amb una plantilla e indicant les jsps o htmls que es carregaran dins les caixes de les plantilles.

Aquesta classe ens ve dins al jar **LWF.jar**.

```
LWF.view.LWFViewFilter.java
```

També tenim una classe on s'emmagatzema aquest mapeig proporcionant aquest servei al filter.

```
LWF.mapping.LWFViewService.java
```

S'ha d'incloure dins el fitxer **web.xml** de l'aplicació a desenvolupar. Per poder validar hem d'emprar o que classe amb els mètodes de validació de LWF o una nostra.

```
<filter>
        <filter-name>view</filter-name>
        <filter-class>LWF.view.LWFViewFilter</filter-class>
```

```
</filter>

<filter-mapping>
        <filter-name>view</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Aquest mapping s'enregistrarà creant un fitxer de tipus propertis al projecte que volem desenvolupar.

```
/aplicació/src/View.xml
```

Exemple de construcció de plantilles amb el fitxer View.xml.

```
<view>
        <page name="pantalla.manteniment.usuari" template="pantalla_manteniment">
                <element name="cap" value="capçalera.jsp"/>
                <element name="ms" value="menusuperior.jsp"/>
                <element name="me" value="menuesquerra.jsp"/>
                <element name="cos" value="manteniment_usuari.jsp"/>
                <element name="peu" value="peu.jsp"/>
        </page>
</view>
```

i18n

Aquesta característiques també s'ha de donar d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>i18n</filter-name>
        <filter-class>LWF.i18n.LWFi18nFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>i18n</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Per poder donar internacionalitzar els textos es necessari emprar-los com identificadors que es substituiran al fitxers properties segons l'idioma seleccionat.

Per cada idioma seleccionat s'ha de crear un fitxer de configuració, i en tots ells hi ha d'haver els mateixos identificadors.

```
i18n_CA.properties
i18n_EN.properties
i18n_ES.properties
```

Dins cada properties tindrem les traduccions dels identificadors per l'idioma concret.

Per exemple.

```
login.titulo=Login
login.usuari=Usuari
login.password=Clau
login.submit=Accedeix

manteniment.titol=Directori
manteniment.logout=Sortir
manteniment.peu=Melchor Vives Bernat - PFC - Enginyeria en Informàtica - UOC
manteniment.missatge=Missatge
```

Per realitzar aquesta traducció d'identificador + idioma al text real s'ha d'invocar un mètode del servei d'i18n. Un exemple.

```
<%@page import="LWF.i18n.LWFi18nService" %>
<%=LWFi18nService.traduce("manteniment.logout")%>
```

1.4 Etapa de creació d'una aplicació d'exemple

1.4.1 Anàlisis de l'exemple

1.4.1.1 Explicació de l'aplicació d'exemple

Una vegada tenim codificat el nostre marc de treball que hem anomenat LWF, només ens falta crear una aplicació web a mode d'exemple per poder demostrar, justament amb el tutorial del LWF, com es crea una aplicació des de cero emprant el marc de treball LWF, com adaptar els fitxers de configuració del nou projecte per el LWF, i finalment, com generar el nou codi a partir de la guia especificada del LWF.

Aquesta aplicació d'exemple ens servirà per mostrar el marc de treball LWF en un entorn real, demostrant les seves qualitats i com les seves característiques i funcionalitats ens ajuden a assolir els objectius del marcs de treball de la capa de presentació.

Després de mesurar quina podria ser l'aplicació d'exemple més adequada al nostra cas, tant per dimensió com per tecnologia, s'ha decidit crear una aplicació web per mantenir una base de dades de servei de directori o LDAP.

Com a anècdota, un dels factors que han influït en aquesta decisió ha estat que LDAP és l'acrònim de Lightweight Directory Access Protocol, i casualment la primera sigla de LDAP igual que les dues primeres de LWF, fan referència a la mateixa paraula: lightweight.

Aquesta aplicació d'exemple l'hem anomenat simplement Directori. La dimensió de l'aplicació no és ni massa gran ni massa petita i te una quantitat suficient de pantalles com per mostrar totes les característiques del marc de treball LWF. Tampoc requereix manteniment massa complexos ni formularis amb molts camps i relacions encadenades amb altres entitats o taules de base de dades. Com que es un backend també podem incloure un login.

1.4.1.2 Llistat de requeriments funcionals

La funcionalitat de l'aplicació és mols senzilla d'entendre.

Primer tenim un login per poder accedir amb els permisos adequats a l'aplicació, en el nostre cas només tenim l'administrador. Tota aplicació backend requereix programar una autentificació per restringir les funcionalitats només a les persones adequades, en cap cas ha de ser pública.

Una vegada hem accedit a l'aplicació tenim tres manteniments.

El primer manteniment es per els usuaris, podem crear nous usuaris al directori. Els usuaris conformen la part més important d'un servei de directori ja que els LDAP es van dissenyar per aportar més velocitat, seguretat i agilitat que les bases de dades relacionals per aquest tipus de consultes.

El segon manteniment es per crear grups, també els podríem anomenat rols. Aquest atribut d'un directori també és molt important ja que un servei de directori pot ser consultat per moltes aplicacions però cada usuari només hauria de poder accedir a algunes d'elles i amb un nivell de permisos configurable gràcies als grups als que pertanyen els usuaris. Una altre característica que permet mostrar l'aplicació és el canvi d'idioma o internacionalització.

Finalment tenim un manteniment per associar els usuaris amb els grups. Qualsevol usuari pot estar a molts grups, i igualment cada grup pot tenir tots els usuaris possibles.

1.4.1.3 Llistat de casos d'ús

Els casos d'us que contempla l'aplicació d'exemple anomenada Directori són els següents.

- Autentificació a la pantalla de login perquè l'usuari pugui accedir a l'aplicació
- Consultar el manteniment d'usuaris
- Consultar el manteniment de grups
- Consultar el manteniment per associar usuaris amb grups
- Crear nous usuaris
- Crear nous grups
- Associar usuaris amb grups
- Sortir de l'aplicació
- Canviar d'idioma

1.4.1.3 Objectius a assolir per emprar el marc de treball LWF

Objectius dels marcs de treball

Un dels principals motius per incloure un marc de treball en el disseny d'una aplicació es que ens aporta un conjunts d'objectius genèrics que ens asseguren una sèrie de millores en el procés i en el producte resultant.

Al introduir un nou marc de treball, com en qualsevol altre element nou, es necessari un temps d'aprenentatge i assimilació, però una vegada superat aquest repte inicial un marc de treball proporciona moltes més avantatges que implementar una aplicació des de cero.

A continuació llistarem alguns d'aquest objectius dels marcs de treball (ja exposats en punts anteriors).

- Disminució del cost del projecte
- Disminució del temps de desenvolupament
- Disminució de les incidències a l'entorn de producció.
- Facilitat d'integració de nous recursos al projecte.

Característiques del nostre marc de treball LWF

Cada marc de treball proporciona unes característiques pròpies que depenen del fi que persegueixen. No és el mateix un marc de treball per la capa de persistència que per la capa de presentació, ni tampoc un marc de treball que implementa totes les capes d'un projecte, ni un marc de treball orientat a components RIA.

El nostre marc de treball LWF proporciona un conjunt de característiques bàsiques, la majoria comunes a tots els marcs de treball de la capa de presentació. Amb aquesta aplicació d'exemple s'ha intentat mostrar com s'implementen i funcionen aquest conjunt de característiques.

A continuació llistarem aquestes característiques (ja exposats en punts anteriors).

- Ús d'estàndards i tecnologies ja consolidades com java, jsp, xml, properties, servlet
- Centralitzar en una controladora les peticions HTTP.
- URLs amigables.
- Gestió automàtica de l'intercanvi d'informació entre client i servidor.
- Validacions automàtiques de les dades del formulari.
- Construir les pàgines web com plantilles.
- Internacionalització dels textos.
- Notificacions dels errors i accions a la capa web.

1.4.2 Disseny de l'exemple

1.4.2.1 Breu explicació del disseny de l'exemple

Com hem comentat abans, aquesta aplicació és un backend per gestionar una base de dades de servei de directori tipus LDAP.

La arquitectura del disseny de l'aplicació és ben senzilla.

Primer tindrem una pantalla de login per poder autenticar-nos a l'aplicació i entrar amb un rol determinat.

Després podrem seleccionar entre tres manteniments diferents. Un manteniment per crear usuaris, una altre per crear grups, i finalment un altre per associar usuaris amb grups.

Cada un d'aquests manteniments tindrà un menú per canviar de pantalla, un formulari per crear nous registres i un llistat per mostrar-ne els actuals.

1.4.2.2 Característiques implementades

Controladora

Tota aplicació web basada en el patró de disseny MVC conté una controladora que

centralitza totes les peticions dels usuaris i les gestiona de forma independent. Per utilitzar la controladora implementada al LWF simplement l'hem de configurar donant-la d'alta al fitxer web.xml.

URLs amigables

Gràcies al mapping que proporciona LWF podem implementar URLs amigables. Les URLs amigables proporcionen avantatges en l'indexació i posicionament als cercadors web, com per exemple Google, a més de major visibilitat als usuaris web. Només hem de configurar el mapping al fitxer web.xml i donar d'alta els paths de les URLs amigables dins el fitxer de configuració Mapping.properties fent referència a les peticions d'usuari implementades en l'aplicació d'exemple.

Formularis

Un altre utilitat que proporciona el marc de treball LWF és l'extracció de les dades del formulari que s'envia en la petició HTTP que l'usuari fa al servidor, i setejarles a una classe accessible per el programador. Simplement tenim que donar d'alta aquesta utilitat en el fitxer web.xml i incloure dins el formulari HTTP un camp hidden amb el nom de la classe java.

Validacions

LWF proporciona la validació automàtica dels formularis HTTP prèviament configurat les restriccions d'aquests camps dins el fitxer de configuració Validacions.xml. A part de les validacions que LWF proporciona per defecte també podem estendre aquesta classe amb noves validacions implementades per nosaltres. Com a la resta d'utilitats també s'ha de donar d'alta dins el fitxer web.xml.

Plantilles

Un altre característica important de les aplicacions web consisteix en construir les pàgines web com plantilles. La majoria de les aplicacions web tenen una estructura similar es va repetint en la major part de la navegació canviant només parts de la mateixa. Les plantilles permeten independitzar totes aquestes parts, millorant la reutilització, la creació de noves pàgines i el manteniment de les mateixes. Al LWF hem de donar d'alta aquesta utilitat al fitxer web.xml i configurar el fitxer de configuració View.xml amb les pàgines de l'aplicació d'exemple.

Internacionalització

Avui en dia pràcticament totes les pàgines web estan internacionalitzades ja que hi poden accedir usuaris de tot el mon i és un mecanisme per capar usuaris per la nostra web i no vagin a una altra perquè no entenen els continguts. Per emprar aquesta utilitat s'ha de donar d'altra al fitxer web.xml i crear un fitxer de configuració per cada idioma que vulguem emprar.

Notificacions

Finalment, en qualsevol aplicació on es genera una comunicació entre l'usuari i el

servidor hem de poder notificar els errors, les alertes i altres informacions a l'usuari. LWF permet gestionar-ho de forma senzilla. Només hem d'utilitzar les operacions que ja ens proporciona la classe implementada al LWF.

1.4.3 Implementació

1.4.3.1 Implementació de l'aplicació d'exemple

Crear el projecte per l'aplicació d'exemple: Directori

Per implementar la nostra aplicació d'exemple Directori hem utilitzat l'IDE Eclipse.

Per el nostre projecte hem creat un Dinamyc Web Project amb el nom de UOC_PFC_PAC4, per diferenciar-lo del que teníem per implementar el marc de treball LWF, anomenat UOC_PFC_PAC3.

Incloure el marc de treball LWF

Una vegada tenim creat el projecte hem d'incloure el marc de treball LWF abans de començar a implementar-lo.

Aquesta acció es tant senzilla com incloure els següents jars:

- LWF.jar
- Jdom.jar

A la carpeta següent.

- /WebContent/WEB-INF/lib/

Incloure els fitxers de configuració per el LWF

Una vegada tenim creat el projecte i inclòs el marc de treball LWF, el pròxim pas serà crear els fitxers de configuració que necessita el marc de treball LWF i que es configuraran per l'aplicació d'exemple. Aquests fitxers són els següents.

- **Mapping.properties**, contindrà les URLs amigables i les relacionarà amb les classes que implementen les accions d'usuari.
- **Validation.xml**, per donar d'alta les validacions automàtiques que es vulguin executar sobre els camps dels formularis HTTP que s'envien al servidor.
- **View.xml**, per donar d'alta noves pàgines web a partir de plantilles.
- **i18n_CA.properties**, per donar d'alta els textos en català que visualitza l'aplicació.
- **i18n_EN.properties**, per donar d'alta els textos en anglès que visualitza l'aplicació.
- **i18n_ES.properties**, per donar d'alta els textos en castellà que visualitza l'aplicació.
- **web.xml**, aquest fitxer és molt important perquè és on donarem d'alta les diverses característiques que proporciona LWF.

Implementació de les característiques del LWF

Quan ens trobem en aquest punt ja ho tenim tot preparat per començar a programar la nostra aplicació.

Per començar podem incloure una nova pàgina web dins els fitxers de configuració.

- La URL amigable dins el *Mapping.properties*, així com la classe que gestionarà l'acció de l'usuari.
- Donem d'alta la pàgina web dins el fitxer *View.xml* i assignem a cada part de la plantilla la seva part corresponent.
- Per els nous textos que sortiran per pantalla, hem d'incloure les traduccions dins del properties internacionalització: *i18n_CA.properties, i18n_EN.properties i i18n_ES.properties*.
- També podem incloure validacions dels camps dels formularis al realitzar peticions HTTP al servidor, només tindrem que incloure els formularis al fitxer *Validation.xml* i indicar per cada camp les operacions o validacions a realitzar.

A part d'aquests fitxers de configuració haurem de crear la classe que implementa l'acció d'usuari, que haurà d'estendre obligatòriament de l'interface *LWFUserRequestInterface*.

Finalment només haurem de crear les jsps que ens facin falta, probablement el *cos.jsp* i *llistat.jsp*.

Controladora

Per emprar aquesta característica només haurem de donar-la d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>controller</filter-name>
        <filter-class>LWF.controller.LWFControllerFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>controller</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

URLs amigables

Per emprar aquesta característica haurem de fer el següent.

Primer, donar-la d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>mapping</filter-name>
        <filter-class>LWF.mapping.LWFMappingFilter</filter-class>
</filter>

<filter-mapping>
```

```
        <filter-name>mapping</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

I després editar el fitxer *Mapping.properties* per incloure el path de la URL amigable.

Per exemple, si tenim la URL http://www.directori.com/balckend/login només hauríem de posar login, perquè ni el domini ni el contexte formen part del path.

El valor del path es la classe que implementa l'acció d'usuari.

```
login = com.directori.peticions.LoginUserRequest
```

Formularis

Per poder copiar els valors del formularis en un classe java hem de fer el següent.

Primer, donar-la d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>formExtractor</filter-name>
        <filter-class>LWF.formExtractor.LWFFormExtractorFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>formExtractor</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Després hem de crear una classe java amb els mateixos valors del formulari HTTP que volem enviar al servidor.

```
com.directori.formularis.AutenticarUserForm
```

Finalment, només ens quedarà incloure un nou camp hidden al formulari indicant-li en nom de la classe on es setejeran aquests valors. Per exemple.

```
<input type="hidden" id="form" name="form"
value="com.directori.formularis.AutenticarUserForm"/>
```

Validacions

Per poder emprar les validacions automàtiques dels camps dels formularis hem de dur a terme les següents accions.

Donar-la d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>validation</filter-name>
```

```
        <filter-class>LWF.validation.LWFValidationFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>validation</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

A part, també li hem d'indicar quina es la classe que implementarà les validacions. LWF ens proporciona la classe LWF.validation.LWFValidationMethods, però per la nostra aplicació d'exemple necessitàvem més validacions, així doncs, hem donat d'alta aquesta classe també al fitxer *web.xml*.

```
<context-param>
         <param-name>validationMethodsClass</param-name>
         <param-value>com.directori.validacions.Validacions</param-value>
</context-param>
```

La classe *com.directori.validacions.Validacions* estén de *LWF.validation.LWFValidationMethods*, d'aquesta manera hereta tots els seus mètodes i a part, hem podem afegir de nous, com el següent.

```
public Boolean isMail(String text)
{
        if (text.matches("\\w+@\\w+\\.\\w+")) {
                return new Boolean(true);
        } else {
                return new Boolean(false);
        }
}
```

Finalment, només ens faltarà configurar les validacions al fitxer *Validation.xml*. A continuació un exemple.

```
<validation>
        <form name="AutenticarUserForm">
                <input name="user" method="isMandatory,isString"/>
                <input name="password" method="isMandatory,isPasswordPattern"/>
        </form>
</validation>
```

Plantilles

Igual que les anteriors característiques també hem de donar d'alta aquesta característica al fitxer *web.xml*.

```
<filter>
        <filter-name>view</filter-name>
        <filter-class>LWF. view. LWFViewFilter</filter-class>
```

```
</filter>

<filter-mapping>
        <filter-name>view</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Per construir les pàgines web, la configuració s'ha d'aplicar al fitxer *View.xml*.

```
<view>
        <page name="pantalla.manteniment.usuaris"
template="plantilles/manteniment.jsp">
                <element name="cap" value="/generic/cap.jsp"/>
                <element name="missatge" value="/generic/missatge.jsp"/>
                <element name="menu" value="/generic/menu.jsp"/>
                <element name="cos" value="/pantalles/usuaris/cos.jsp"/>
                <element name="llistat" value="/pantalles/usuaris/llistat.jsp"/>
                <element name="peu" value="/generic/peu.jsp"/>
        </page>
</view>
```

Per acabar amb aquesta característica només ens faltaria incloure un tag especial a les jsp que siguin de tipus plantilla. Aquest tag llegeix de sessió la jsp que volem incloure, sempre començant per "TEMPLATE_" seguit del nom de l'element que hem configurat dins el fitxer View.xml. Per exemple.

```
<% String cap = (String) request.getAttribute("TEMPLATE_cap"); %>
<jsp:include page="<%=cap%>"></jsp:include>
```

Internacionalització

Aquesta característiques també s'ha de donar d'alta al fitxer *web.xml*.

```
<filter>
        <filter-name>i18n</filter-name>
        <filter-class>LWF.i18n.LWFi18nFilter</filter-class>
</filter>

<filter-mapping>
        <filter-name>i18n</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Per poder donar internacionalitzar els textos es necessari emprar-los com identificadors que es substituiran al fitxers properties segons l'idioma seleccionat.

Per cada idioma seleccionat s'ha de crear un fitxer de configuració, i en tots ells hi ha d'haver els mateixos identificadors.

```
i18n_CA.properties
i18n_EN.properties
i18n_ES.properties
```

Dins cada properties tindrem les traduccions dels identificadors per l'idioma concret. Per exemple.

```
login.titulo=Login
login.usuari=Usuari
login.password=Clau
login.submit=Accedeix

manteniment.titol=Directori
manteniment.logout=Sortir
manteniment.peu=Melchor Vives Bernat - PFC - Enginyeria en Informàtica - UOC
manteniment.missatge=Missatge
```

Per realitzar aquesta traducció d'identificador + idioma al text real s'ha d'invocar un mètode del servei d'i18n. Un exemple.

```
<%@page import="LWF.i18n.LWFi18nService" %>
<%=LWFi18nService.traduce("manteniment.logout")%>
```

Notificacions

Les notificacions són un servei que proporciona la següent classe.

```
LWF.message.LWFMessageService.java
```

Per enviar una notificació des del servidor a una pàgina web són necessàries dues accions.

Primer, des del servidor hem d'invocar un dels dos mètodes per enviar el missatge.

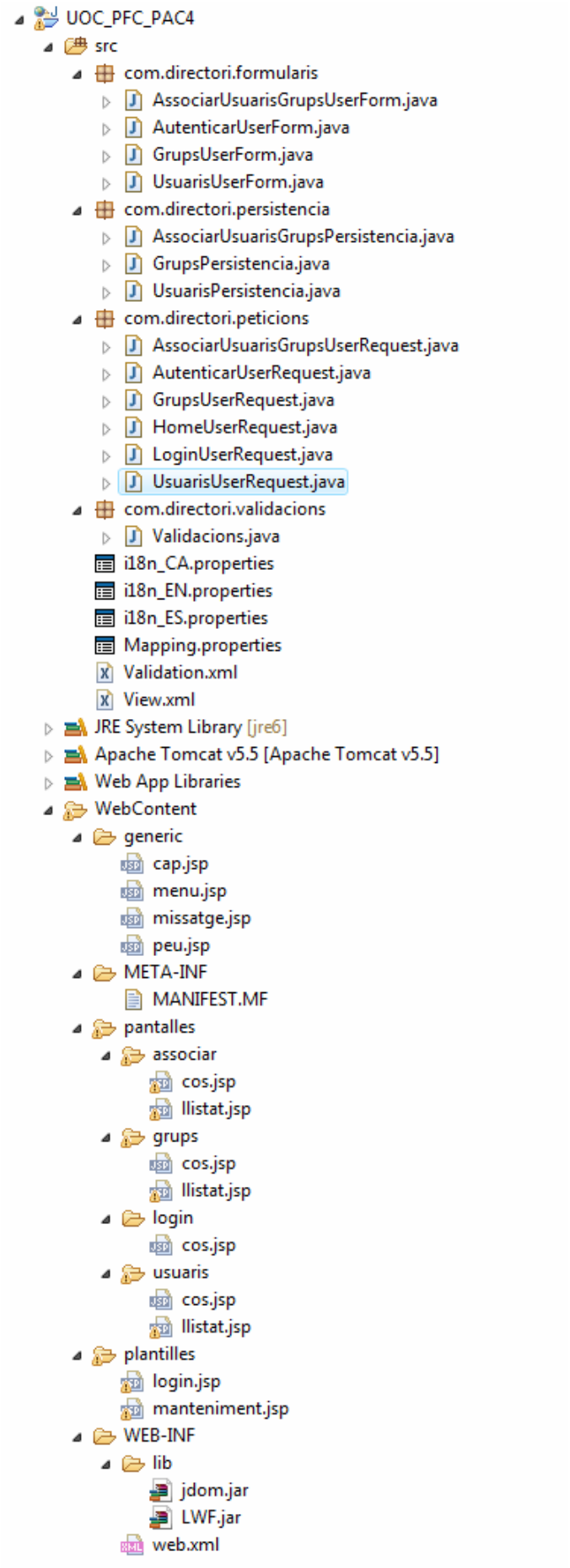
```
LWFMessageService.addInfo("missatge.login.ok");
LWFMessageService.addError("missatge.login.error");
```

Finalment, hem de recuperar aquest valor de l'interfície web i ho podem fer amb una estructura semblant a la següent.

```
<%
if (LWFMessageService.existe(request)) {
%>
        ...
        <%= LWFMessageService.printInHTML(request) %>
        ...
<%
}
%>
```

Estructura final del nostra projecte

Aquesta és l'estructura final del projecte per generar la nostra aplicació web d'exemple.


UOC_PFC_PAC4
src
com.directori.formularis
AssociarUsuarisGrupsUserForm.java
AutenticarUserForm.java
GrupsUserForm.java
UsuarisUserForm.java
com.directori.persistencia
AssociarUsuarisGrupsPersistencia.java
GrupsPersistencia.java
UsuarisPersistencia.java
com.directori.peticions
AssociarUsuarisGrupsUserRequest.java
AutenticarUserRequest.java
GrupsUserRequest.java
HomeUserRequest.java
LoginUserRequest.java
UsuarisUserRequest.java
com.directori.validacions
Validacions.java
i18n_CA.properties
i18n_EN.properties
i18n_ES.properties
Mapping.properties
Validation.xml
View.xml
JRE System Library [jre6]
Apache Tomcat v5.5 [Apache Tomcat v5.5]
Web App Libraries
WebContent
generic
cap.jsp
menu.jsp
missatge.jsp
peu.jsp
META-INF
MANIFEST.MF
pantalles
associar
cos.jsp
llistat.jsp
grups
cos.jsp
llistat.jsp
login
cos.jsp
usuaris
cos.jsp
llistat.jsp
plantilles
login.jsp
manteniment.jsp
WEB-INF
lib
jdom.jar
LWF.jar
web.xml

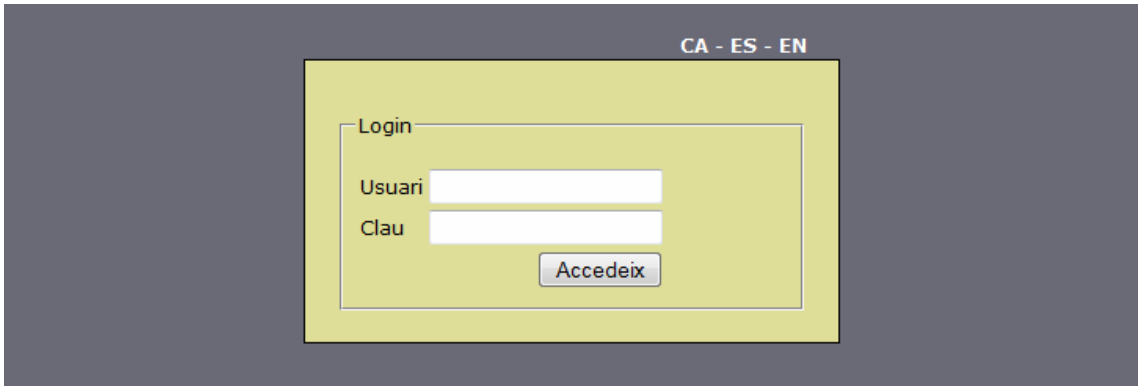
1.4.3.2 Presentació de l'aplicació d'exemple

Pantalla login

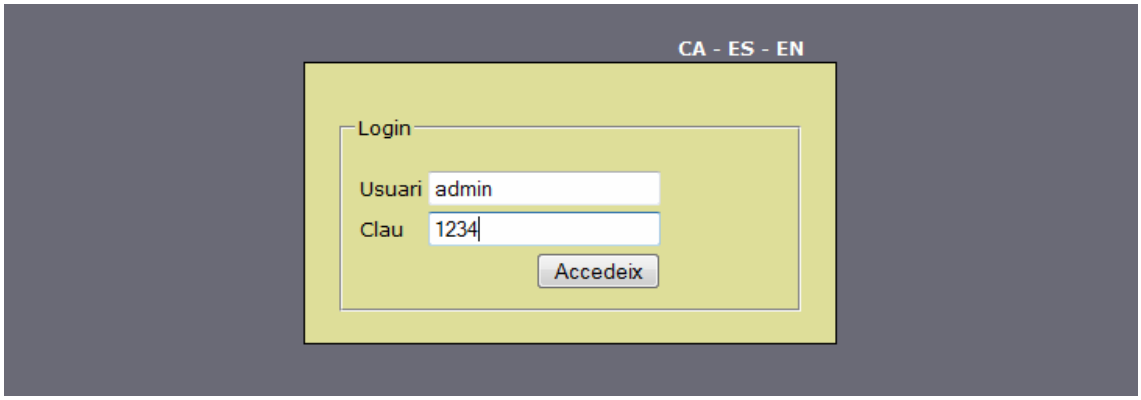
Al accedir a l'aplicació mitjançant la següent URL http://localhost:8080/directori/login s'ens presenta la pantalla de login per poder realitzat l'autentificació i entrar a l'aplicació.

Simplement hem d'insertar l'usuari i password requerit.

A la pantalla de login també tenim els enllaços per canviar d'idioma, per aquesta aplicació tenim CA = català, ES = castellà i EN = anglès.

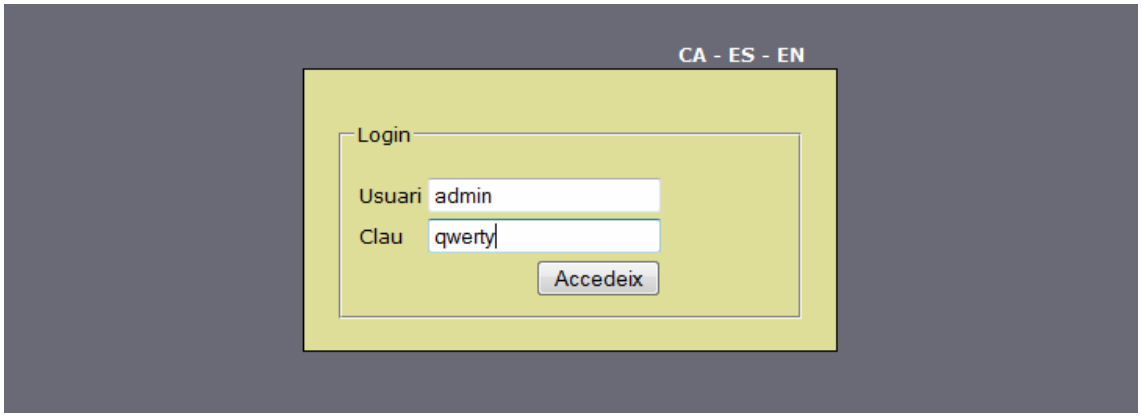


Per poder realitzar l'autentificació satisfactòriament és necessari introduir les següents dades.

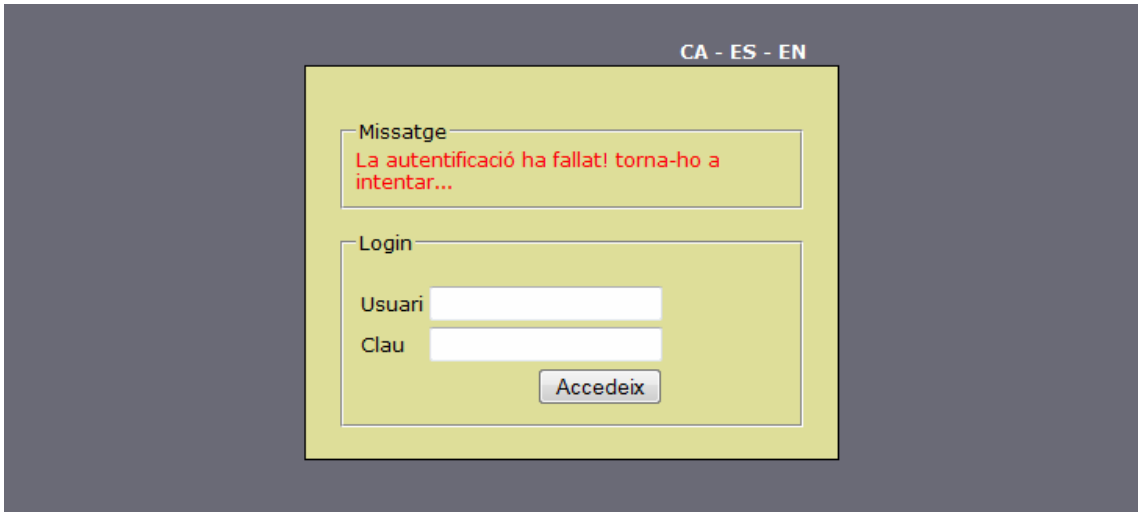


Si l'autentificació ha sigut correcta entrarem a l'aplicació, s'ens notificarà amb un missatge que hem entrat correctament i ja podrem accedir a tots els manteniments que aquesta proporciona.

En el cas que introduïm dades incorrectes al formulari per l'autentificació.



Si l'autentificació ha sigut incorrecta s'ens mostrarà un missatge d'error a la mateixa pantalla de login.



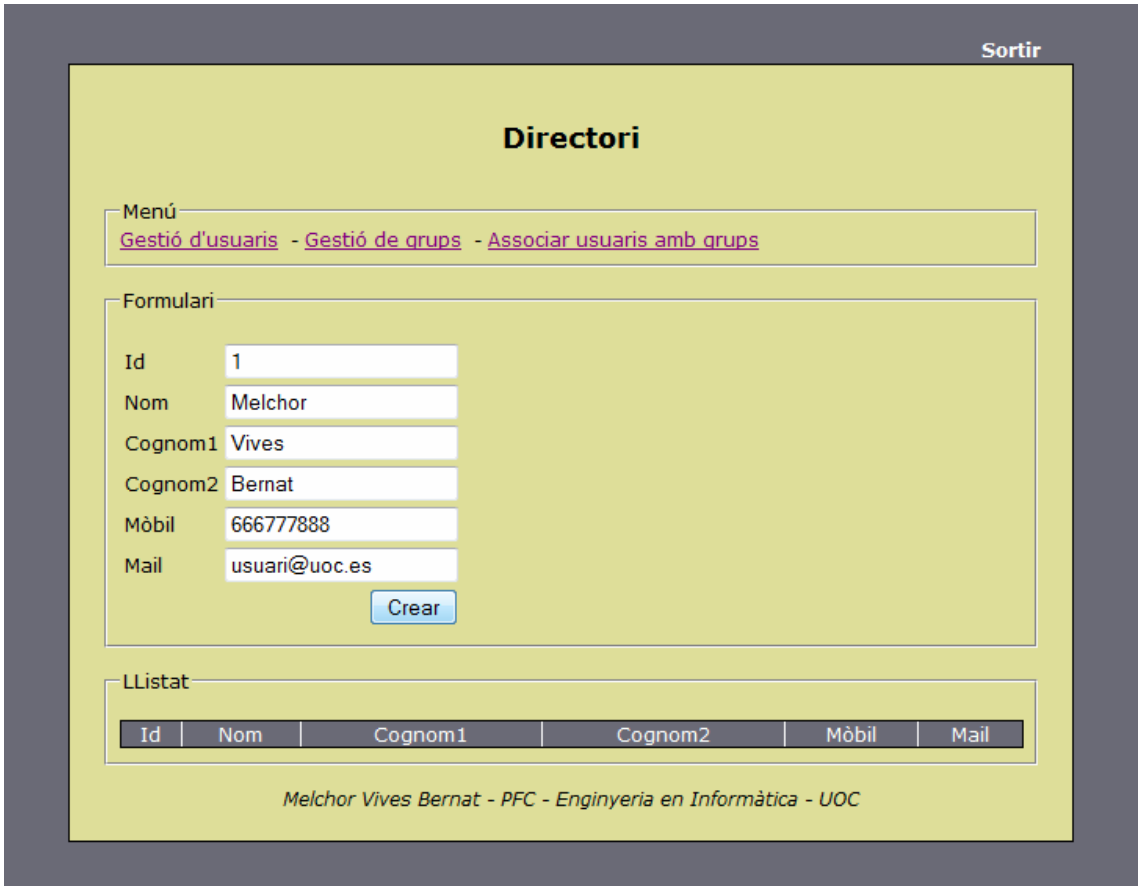
Manteniment d'usuaris

Una vegada ens hem autenticar podem accedir a la pantalla d'usuaris.

En aquesta pantalla es podem crear nous usuaris i veure en el llistat els usuaris existents actualment.

Les dades dels usuaris són mínimes i serveixen purament per mostrar el flux de dades i les validacions.

Podem crear un nou usuari introduint les dades i submitant el formulari amb el botó Crear.



Cada vegada que creem un nou usuari podem visualitzar-lo al llistat d'usuaris.

Manteniment de grups

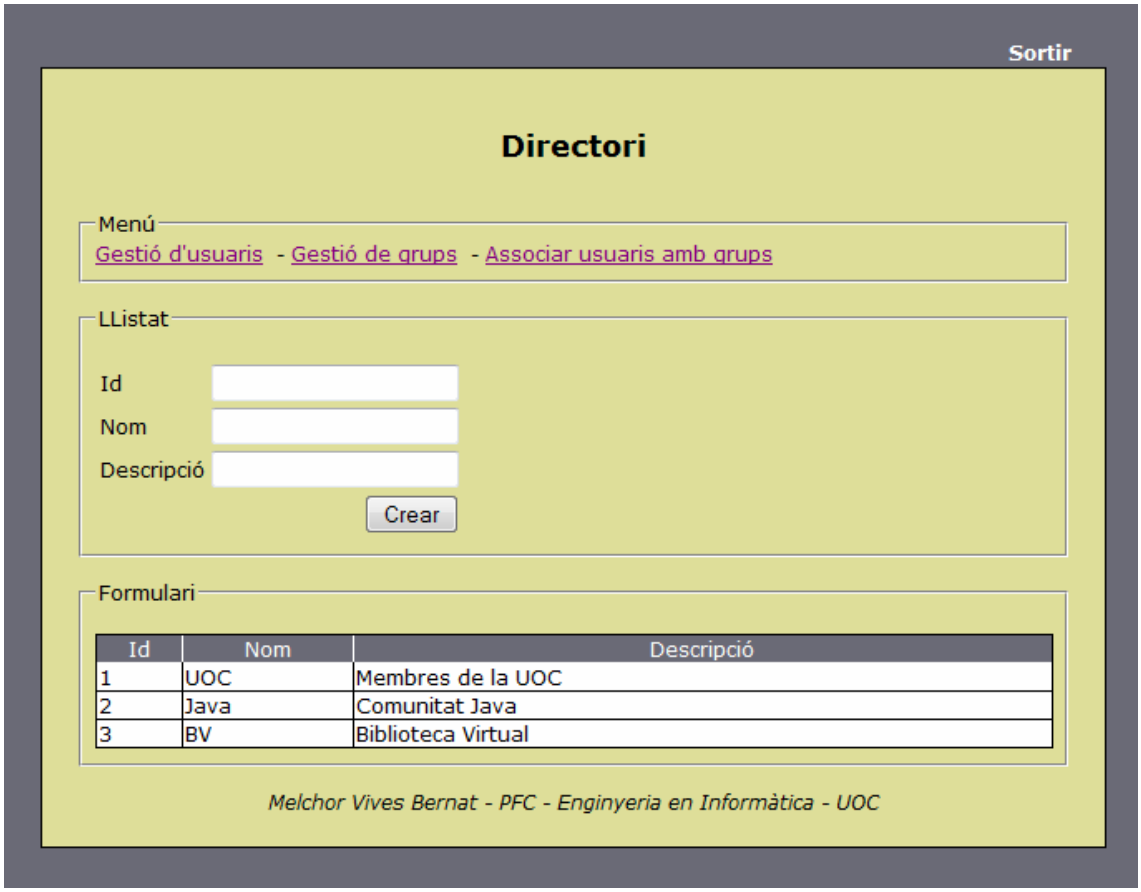
Un altre opció que tenim és accedir a la pantalla de grups.

En aquesta pantalla es podem crear nous grups i veure en el llistat els grups existents actualment.

Les dades dels grups són mínimes i serveixen purament per mostrar el flux de dades i les validacions.

Podem crear un nou grup introduint les dades i submitant el formulari amb el botó Crear.

Cada vegada que creem un nou grup podem visualitzar-lo al llistat de grups.

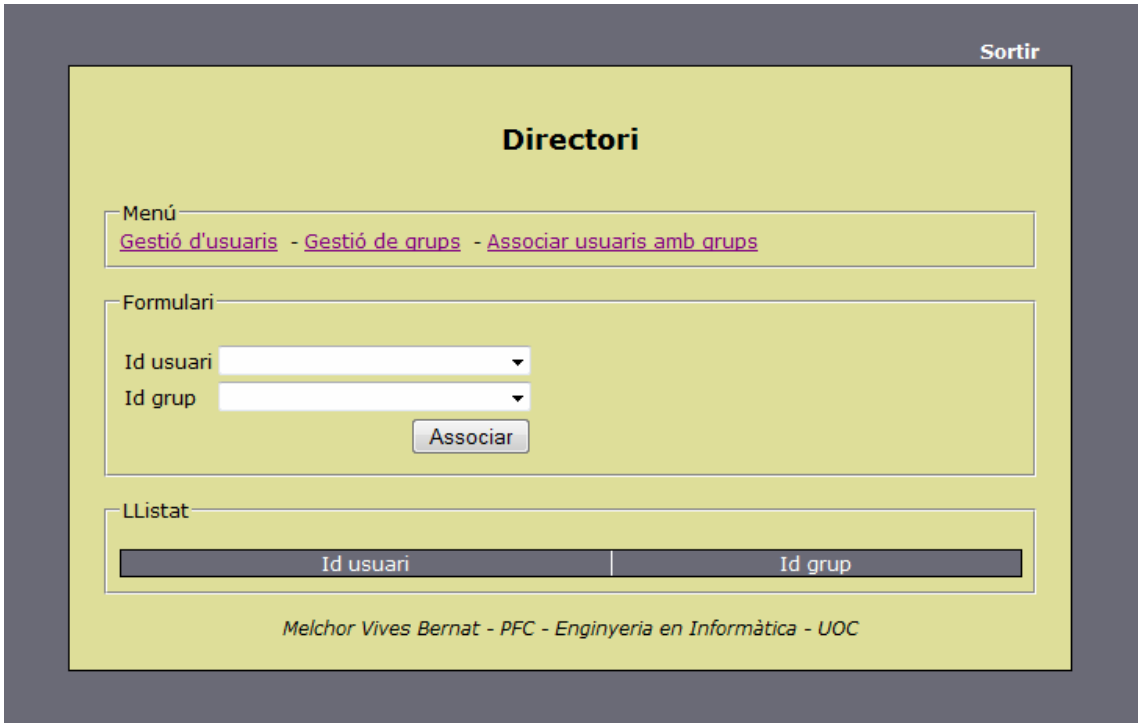


Manteniment per associar usuaris amb grups

Finalment, també tenim una pantalla per associar els usuaris amb els grups.

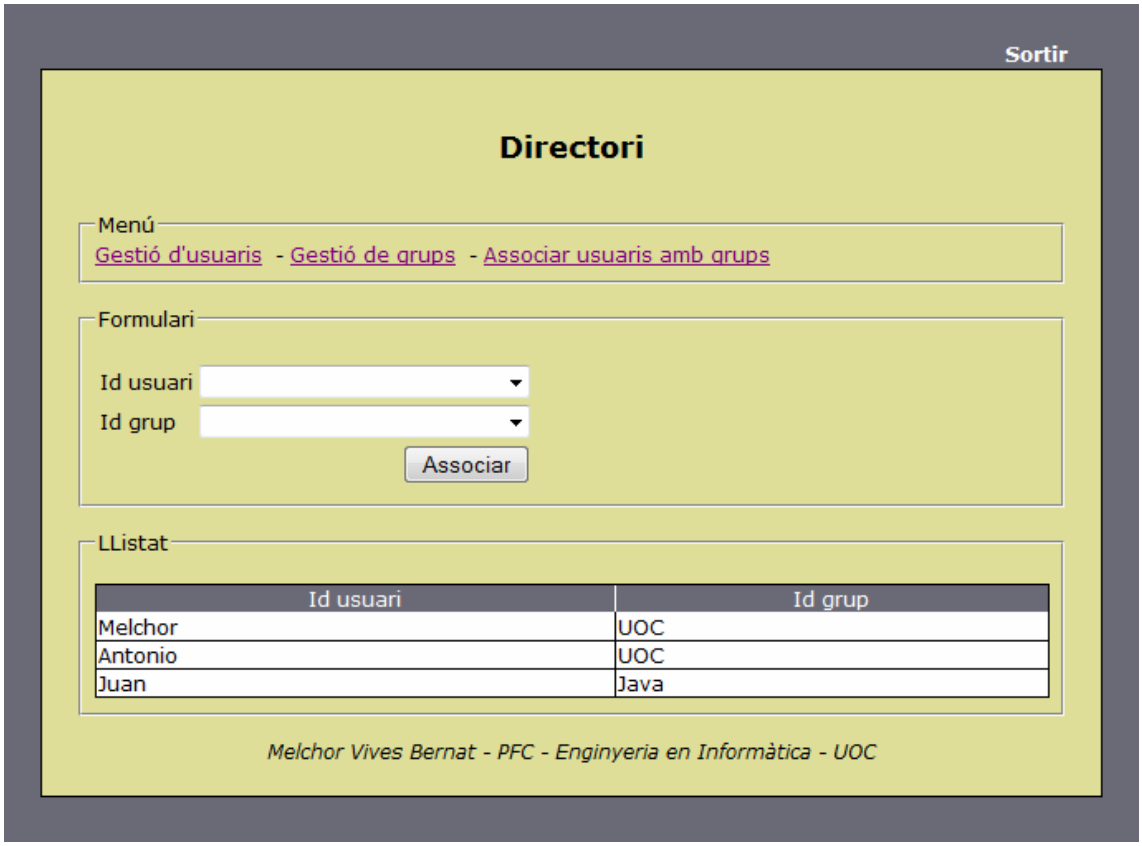
En aquesta pantalla es podem crear noves associacions entre usuaris i grups i veure en el llistat les associacions existents actualment.

Les dades de l'associació són mínimes i serveixen purament per mostrar el flux de dades i les validacions.



Per crear una nova associació, s'ha de seleccionar l'usuari i el grup dins dos seleccionables que contenen tots els registres existents per els dos casos. Els dos camps són obligatoris, i una vegada seleccionats hem de submitar el formulari amb el botó Associar.

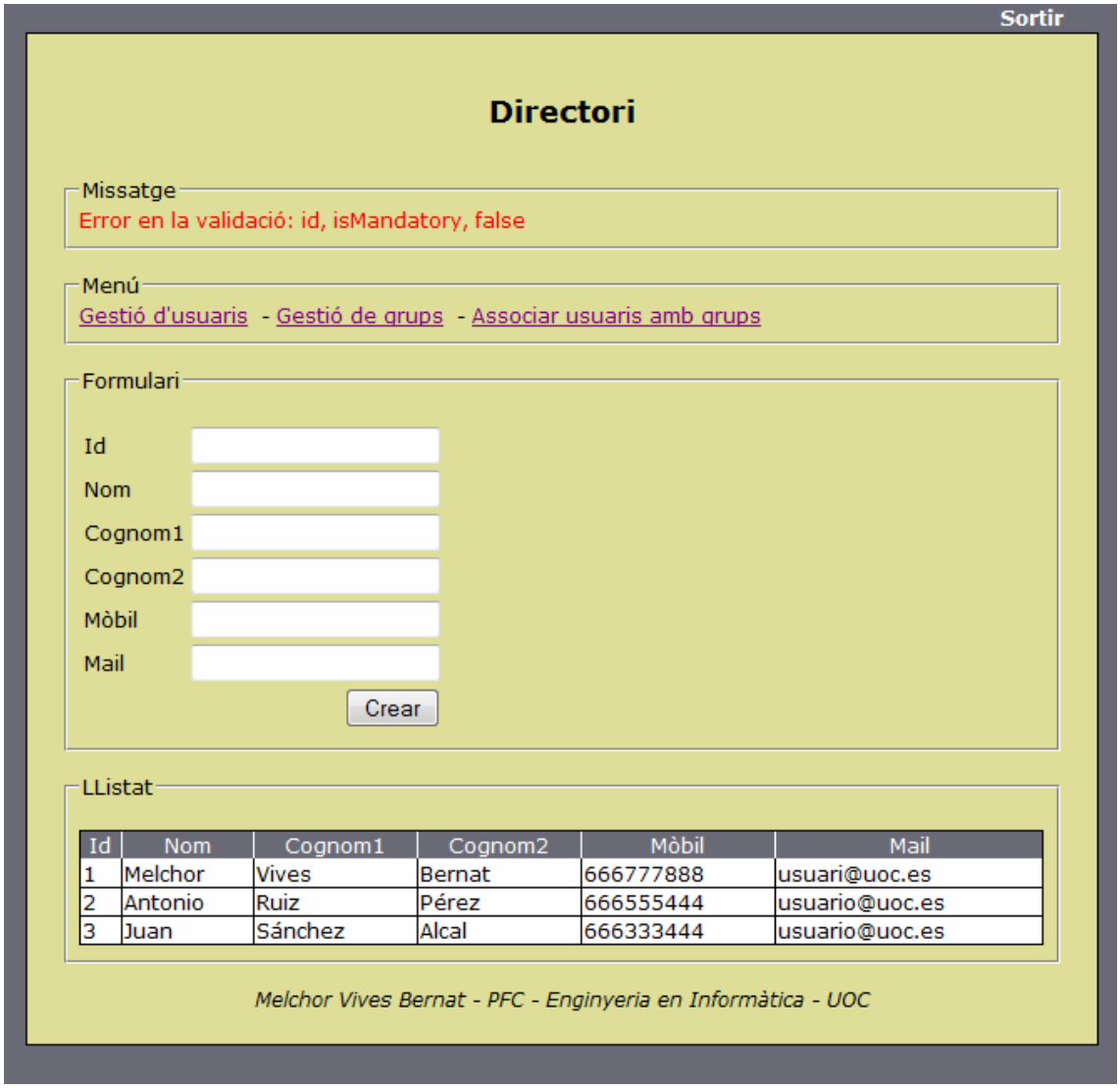
Cada vegada que creem una nova associació podem visualitzar-la al llistat.



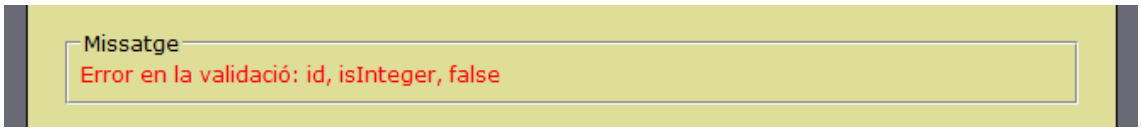
Altres validacions

A qualsevol dels tres manteniments, al intentar crear nous registres s'executa la validació automàtica i si no es compleix alguna d'aquestes operacions es retorna un missatge d'error a la pàgina web.

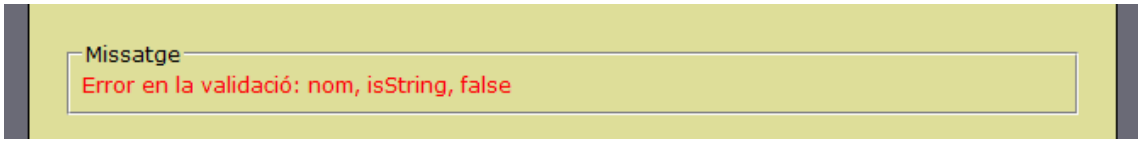
En aquest cas s'ha intentat crear un usuari sense indicar les dades obligatòries del formulari, concretament el camp Id.



En aquest cas, s'ha intentat crear un usuari introduint un text al camp Id quan s'està esperant un nombre enter.



En aquest cas, s'ha intentat crear un usuari introduint un nombre enter al camp Nom quan s'està esperant un String.



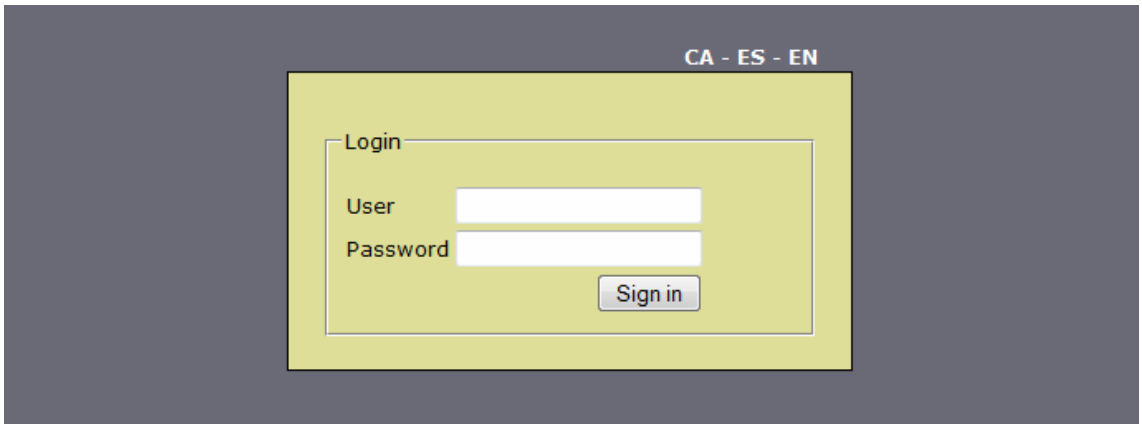
Canvi d'idioma

Una de les característiques implementades és l'internacionalització de l'aplicació web.

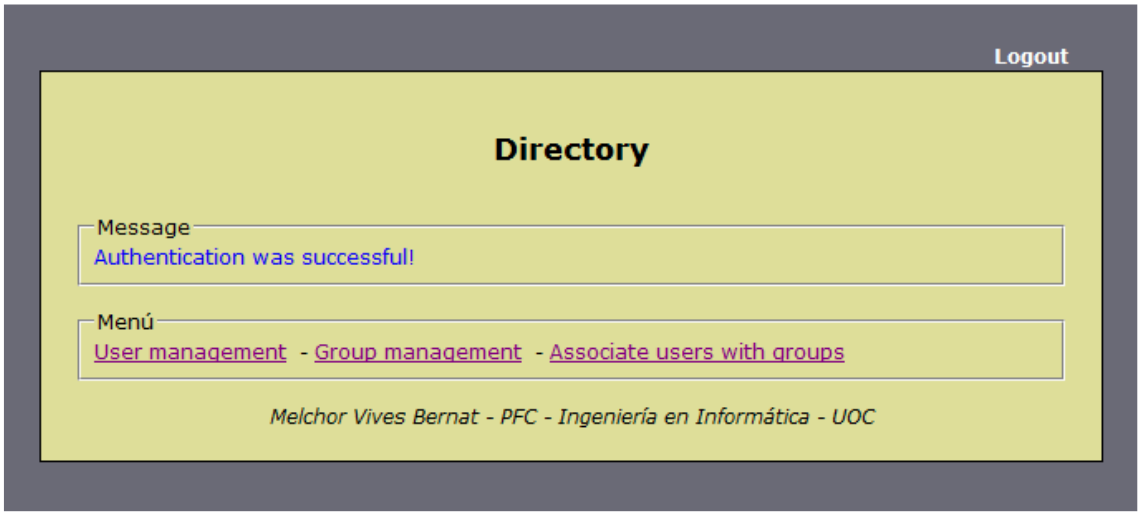
Per canviar d'idioma, s'ha decidit implementar aquesta opció només a la pantalla de login. Per fer-ho tenim disponibles els enllaços dels idiomes implementats i suportats per l'aplicació.

Els enllaços són els següents: CA = català, ES = castellà i EN = anglès.

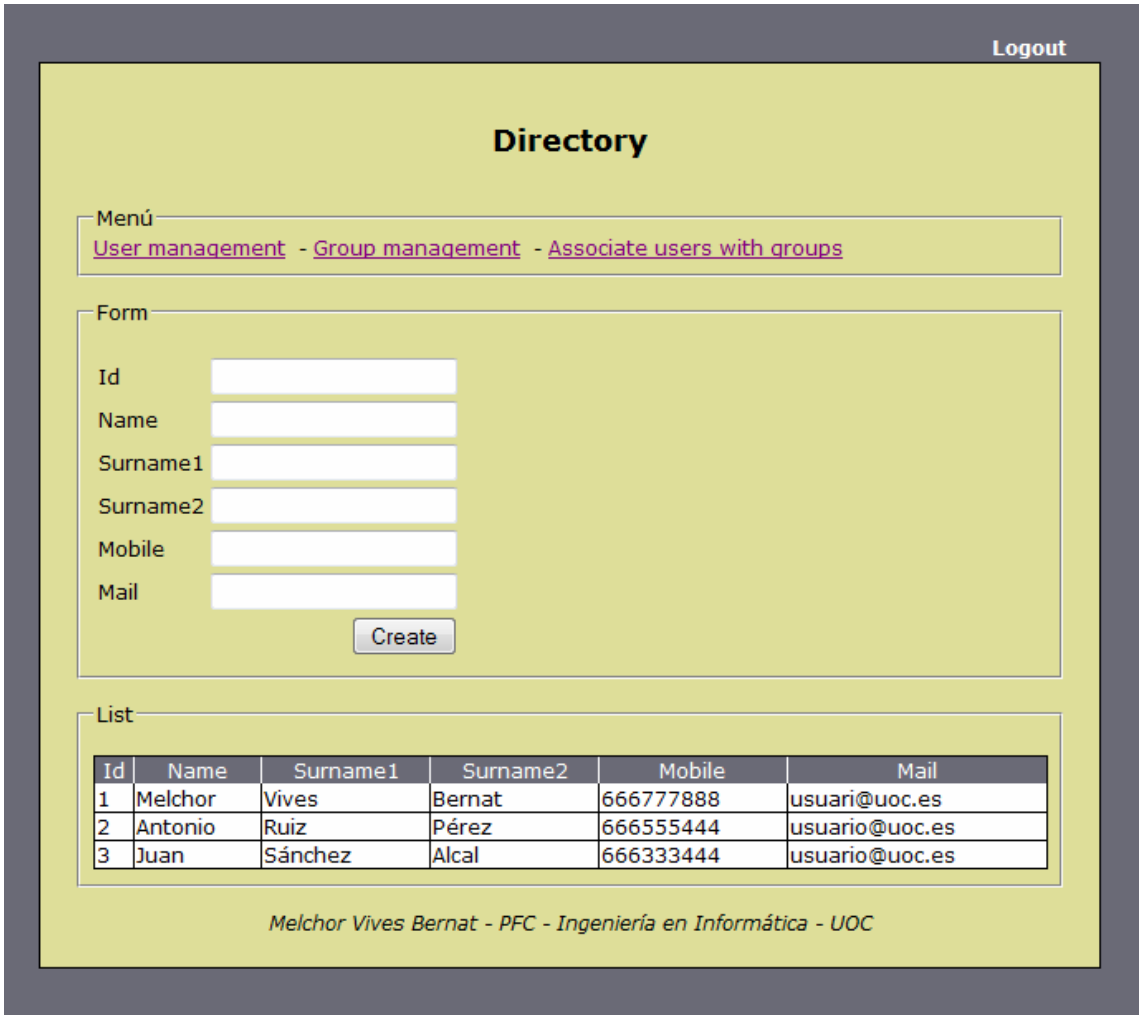
Ara mostrarem una navegació amb anglès.



Un exemple de la home en anglès.



Un exemple del manteniment d'usuaris en anglès.



Si ja estàs dins l'aplicació, si vols tornar a la pantalla de login s'ha de clickar a Sortir.

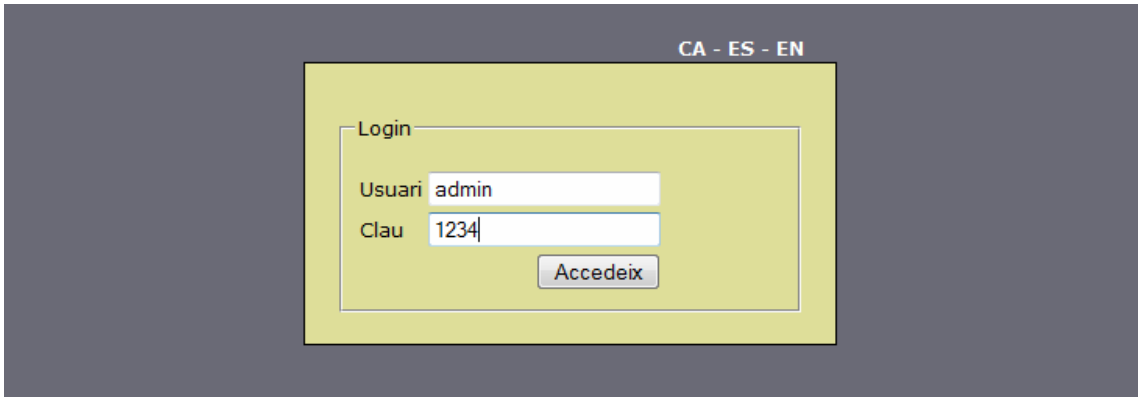
1.4.3.3 Altres consideracions

Usuari i password

Per poder accedir a l'aplicació i realitzar l'autentificació satisfactòriament és necessari introduir les següents dades.

- Usuari = admin
- Clau = 1234

Aquesta és l'única autentificació possible programada per l'aplicació d'exemple.



Sense BBDD

Per simplificar la programació de l'aplicació d'exemple s'ha evitat emprar una base de dades per guardar l'informació generada.

S'ha decidit així, perquè surt de l'àmbit de la demostració, aporta complexitat, no millora l'objectiu de l'exemple i requereix l'instal·lació d'un nou programa per executar l'aplicació d'exemple.

Per donar suport a la persistència de l'informació, la solució aportada consisteix en guardar les dades en memòria estàtica mitjançant entitats per guardar les dades i classes Hashmap per guardar i recuperar per clau més ràpidament aquests registres.

Build.xml per LWF.jar

Per generar més ràpidament el jar del marc de treball s'ha creat un fitxer de tipus ant.

Aquest fitxer s'anomena *build.xml* i el podem trobar a l'arrel del projecte *UOC_PFC_PAC3*.

Al executar aquest fitxer es genera el fitxer *LWF.jar* que contindrà les classes del nostre marc de treball LWP, i aquest fitxer és el que s'ha d'importar a la carpeta */lib* de les noves aplicacions a desenvolupar.

Build.xml per Directori.war

Per generar més ràpidament el war de l'aplicació d'exemple s'ha creat un fitxer de tipus ant.

Aquest fitxer s'anomena *build.xml* i el podem trobar a l'arrel del projecte *UOC_PFC_PAC4*.

Al executar aquest fitxer es genera el fitxer *Directori.war* seguint l'estàndard Java EE. Podem desplegar aquest fitxer en servidors com el Tomcat o el Jboss per executar l'aplicació d'exemple Directori.

Glossari

ADF Oracle Application Development Framework, o simplement Oracle ADF, és un framework comercial de Java per a la creació d'aplicacions empresarials. És una eina del tipus RAD que es basa en patrons de disseny llestos per usar. Proveeix un enfocament visual i declaratiu per al desenvolupament d'aplicacions J2EE.

CGI Common Gateway Interface, va ser una de les primeres tecnologies de la World Wide Web per crear contingut dinàmic per a les pàgines web. En una aplicació CGI, el servidor web passa les sol·licituds del client a un programa extern que normalment empra llenguatges d'script, finalment es retorna al client contingut estàtic tradicional.

Framework Conjunt estandarditzat de conceptes, pràctiques i criteris per enfocar una problemàtica particular. Serveix com a referència per afrontar i resoldre nous problemes d'índole similar.

Grails Framework lliure per aplicacions web desenvolupat sobre el llenguatge de programació Groovy (el qual es basa en Java). L'objectiu de Grails és ser un marc de treball altament productiu seguint paradigmes com ara convenció sobre configuració o no et repeteixis (DRY), proporcionant un entorn de desenvolupament estandarditzat i ocultant gran part dels detalls de configuració al programador.

GWT Google Web Toolkit és un marc de treball creat per Google que permet amagar la complexitat de diversos aspectes de la tecnologia AJAX. El codi es crea en Java i el compilador ho tradueix a HTML i JavaScript, que pot ser ofuscat per optimitzar el rendiment. A més te dos modes d'execució: mode host, s'executa dins la JVM; mode web, només al client.

Java EE Java Enterprise Edition, és una plataforma de programació per desenvolupar i executar software d'aplicacions en el llenguatge de programació Java amb arquitectura de N capes distribuïdes i que es basa amb components de software modulars executant-se sobre un servidor d'aplicacions.

J2EE Nom antic emprat fins la versió 1.4, ara es coneix com Java EE.

Java Llenguatge de programació orientat a objectes, desenvolupat per Sun Microsystems a començaments dels anys 90. El llenguatge en si mateix pren molta de la seva sintaxi de C i C++, però té un model d'objectes més simple i elimina eines de baix nivell, que acostumen a induir a molts errors, com la manipulació directa de punters o memòria. Les aplicacions Java estan típicament compilades en un bytecode que és normalment interpretat o compilat a codi natiu per a l'execució.

JSF JavaServer Faces és una tecnologia i framework per a aplicacions Java basades en web que simplifica el desenvolupament d'interfícies d'usuari en Java EE. JSF usa JavaServer Pages (JSP) com la tecnologia que permet fer el desplegament de les pàgines, però també es pot acomodar a altres tecnologies com XUL.

JSP JavaServer Pages és una tecnologia Java que permet generar contingut dinàmic per

a web, en forma de documents HTML, XML o d'un altre tipus. Aquesta tecnologia és un desenvolupament de la companyia Sun Microsystems.Les JSP's permeten l'utilització de codi Java mitjançant scripts. A més a més, és possible utilitzar algunes accions JSP definides anticipadament mitjançant etiquetes. Aquestes etiquetes poden ser enriquides mitjançant l'utilització de Biblioteques d'Etiquetes (TagLibs o Tag Libraries) externes i també personalitzades.

MVC Model Vista Controlador és un patró d'arquitectura de software que separa les dades d'una aplicació, l'interfície d'usuari, i la lògica de control en tres components distints. El patró es veu freqüentment en aplicacions web, on la vista és la pàgina HTML i el codi que proveeix de dades dinàmiques a la pàgina. El model és el Sistema de Gestió de Base de Dades i la Lògica de negoci, i el controlador és el responsable de rebre els esdeveniments d'entrada des de la vista.

Play! Marc de treball web fet per a Java que ens permet fer moltes coses seguint els paradigmes d'altres marcs de treball de desenvolupament web àgil, entre els quals entren les idees d'altres llenguatges com Python, Ruby o Groovy.

RAD Rapid Application Development, és un terme que donà James Martin en 1991, als processos de desenvolupament d'aplicacions que permeten tenir una aplicació funcionant en un període curt de temps, front als mètodes que segueixen el cicle de vida tradicional per al desenvolupament d'aplicacions.

RIA Les aplicacions Rich Internet Applications són un nou tipus d'aplicacions amb més avantatges que les tradicionals aplicacions Web. Aquesta sorgeix com una combinació dels avantatges que ofereixen les aplicacions Web i les aplicacions tradicionals.

RichFaces Biblioteca open source de components Ajax per JavaServer Faces, organitzada per JBoss.org. Permet una fàcil integració de les capacitats d'Ajax en el desenvolupament d'aplicacions empresarials Java EE.

Servlet Classe Java executada per un servidor d'aplicacions i que respon a invocacions HTTP, servint pàgines dinàmiques. Un objecte Servlet és capaç de rebre una invocació i generar una resposta en funció de les dades de l'invocació, de l'estat del propi sistema i les dades a què pugui accedir.

Smalltalk Sistema informàtic que permet realitzar tasques de computació mitjançant l'interacció amb un entorn d'objectes virtuals. Metafòricament, es pot considerar que un Smalltalk és un món virtual on viuen objectes que es comuniquen mitjançant l'enviament de missatges.

Struts Marc de treball per donar suport al desenvolupament d'aplicacions web baix el patró MVC baix la plataforma Java EE. Struts es va desenvolupar com a part del projecte Jakarta de l'Apache Software Fundation, però actualment es un projecte independent conegut com Apache Struts. Una de les característiques principals d'Struts es reduir el temps de desenvolupament de les aplicacions.

Xerox Xerox Corporation és el proveïdor més gran del món de fotocopiadores de tòner (tinta seca) i els seus accessoris. La seua central general està a Stamford,

Connecticut, encara que la major part de la companyia està situada prop de Rochester, Nova York, on va ser fundada.

ZK Marc de treball per aplicacions web en AJAX, completament en Java, de programari de codi obert, que permet una completa interfície d'usuari per a aplicacions web sense utilitzar JavaScript i amb poca programació.

Bibliografia

http://en.wikipedia.org/wiki/Apache_Struts
http://en.wikipedia.org/wiki/Category:Web_application_frameworks
http://en.wikipedia.org/wiki/Play_Framework
http://en.wikipedia.org/wiki/RichFaces
http://en.wikipedia.org/wiki/WebWork
http://es.debugmodeon.com/articulo/introduccion-a-play-framework-web-java
http://es.scribd.com/doc/97147/introduccion-al-framework-struts
http://es.wikipedia.org/wiki/Categor%C3%ADa:Frameworks_de_Java
http://es.wikipedia.org/wiki/Common_Gateway_Interface
http://es.wikipedia.org/wiki/Framework
http://es.wikipedia.org/wiki/Google_Web_Toolkit
http://es.wikipedia.org/wiki/Grails
http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29
http://es.wikipedia.org/wiki/Java_EE
http://es.wikipedia.org/wiki/Java_Servlet
http://es.wikipedia.org/wiki/JavaServer_Faces
http://es.wikipedia.org/wiki/JavaServer_Pages
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
http://es.wikipedia.org/wiki/Oracle_Application_Development_Framework
http://ca.wikipedia.org/wiki/Rapid_Application_Development
http://es.wikipedia.org/wiki/Rich_Internet_Application
http://es.wikipedia.org/wiki/Smalltalk
http://es.wikipedia.org/wiki/Struts
http://es.wikipedia.org/wiki/Xerox
http://es.wikipedia.org/wiki/ZK_Framework
http://heim.ifi.uio.no/~trygver/
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=miPrimeraWebStruts
http://www.javabeat.net/articles/68-introductiion-to-jakarta-struts-2.html
http://www.javamexico.org/blogs/wishmaster77/productividad_rendimiento_java_play_framework_experiencia_personal
http://www.programacion.com/articulo/manual_basico_de_struts_156
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
http://www.alzado.org/articulo.php?id_art=355
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ajax.devguide.help/docs/PureAjax_MVC_patterns.html
http://code.google.com/p/cemicursoj2ee/wiki/MVCs

Annexes

1.1 LWF

1.1.1 Classes

7.1.1.1 LWFControllerFilter.java

```
package LWF.controller;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import LWF.util.LogUtil;

public class LWFControllerFilter implements Filter
{
	@Override
	public void init(FilterConfig arg0) throws ServletException {
		LogUtil.print("LWPControllerFilter.init()");
	}

	@Override
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
	throws IOException, ServletException
	{
		HttpServletRequest userRequest = (HttpServletRequest) request;
		String path = userRequest.getServletPath().substring(1);
		request.setAttribute("LWF_PATH", path);
		LogUtil.print(" Se ha subido al request el atributo: LWF_PATH="+path);
		chain.doFilter(request, response);
	}

	@Override
	public void destroy() {
		LogUtil.print("LWPControllerFilter.destroy()");
	}
}
```

7.1.1.2 LWFFormExtractorFilter.java

```
package LWF.formExtractor;

import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Enumeration;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import LWF.util.LogUtil;
```

```
public class LWFFormExtractorFilter implements Filter
{
	@Override
	public void init(FilterConfig arg0) throws ServletException {
		LogUtil.print("LWFFormExtractorFilter.init()");
	}

	@SuppressWarnings({ "rawtypes", "unchecked" })
	@Override
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
	throws IOException, ServletException
	{
		LogUtil.print("LWFFormExtractorFilter");

		HttpServletRequest userRequest = (HttpServletRequest) request;

		String form = userRequest.getParameter("form");
		if (form!=null) {

			Enumeration ePar = userRequest.getParameterNames();
			try {
			// Clase
			Class c = Class.forName(form);
			Object o = c.newInstance();

			// Parametros del metodo
			Class[] p = new Class[1];
			p[0] = String.class;

			// Recorremos los parametros
			while (ePar.hasMoreElements()) {
			 try {

				// Recuperamos el parametro y su valor del formulario
				String parametro = (String) ePar.nextElement();
				String valor = userRequest.getParameter(parametro);

				// Metodo
				String metodo = "set"+parametro.substring(0,
1).toUpperCase()+parametro.substring(1);
				Method m = c.getMethod(metodo,p);

				// Ahora los valores que pasaremos a los parametros
				 Object[] input = {valor};

				 // Lo invocamos
					m.invoke(o, input);

				} catch (IllegalArgumentException e) {
					//e.printStackTrace();
				} catch (IllegalAccessException e) {
					//e.printStackTrace();
				} catch (InvocationTargetException e) {
					//e.printStackTrace();
				} catch (SecurityException e) {
					//e.printStackTrace();
				} catch (NoSuchMethodException e) {
					//e.printStackTrace();
				}
			}

		      // Subimos el formulario al request para poder utilizarlo en el userRequest
			request.setAttribute("LWF_FORM", o);

			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			} catch (InstantiationException e) {
				e.printStackTrace();
			} catch (IllegalAccessException e) {
				e.printStackTrace();
			}
```

```
		}
		//LogUtil.print("Se ha subido al request el atributo: LWF_PATH="+path);
		chain.doFilter(request, response);
	}

	@Override
	public void destroy() {
		LogUtil.print("LWFFormExtractorFilter.destroy()");
	}
}
```

7.1.1.3 LWFi18nFilter.java

```
package LWF.i18n;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import LWF.util.LogUtil;

public class LWFi18nFilter implements Filter
{
	@Override
	public void init(FilterConfig arg0) throws ServletException {
		LogUtil.print("LWPi18nFilter.init()");
	}

	@Override
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
	throws IOException, ServletException
	{
		HttpServletRequest userRequest = (HttpServletRequest) request;
		String idioma = userRequest.getParameter("idioma");

		// Si es el primer accés aplicam el primer idioma accesible
		if (LWFi18nService.getIdiomaActivo()==null)
LWFi18nService.setIdiomaActivo(LWFi18nService.getIdiomasAccesiblesCodigo()[0]);

		// Si ens ve l'idioma l'actualitzam
		if (idioma!=null) LWFi18nService.setIdiomaActivo(idioma);

		chain.doFilter(request, response);
	}

	@Override
	public void destroy() {
		LogUtil.print("LWPi18nFilter.destroy()");
	}
}
```

7.1.1.4 LWFi18nService.java

```
package LWF.i18n;

import java.io.IOException;
import java.util.Properties;

public class LWFi18nService
{
	private static final String[] idiomasAccesiblesTexto = {"Català","Español","English"};
	private static final String[] idiomasAccesiblesCodigo = {"CA","ES","EN"};
```

```
	private static String idiomaActivo = null;

	public static String traduce(String texto)
	{
		if (idiomaActivo==null) {
			return "¿¿¿"+texto+"???";
		} else {
		    ClassLoader classLoader = LWFi18nService.class.getClassLoader();
		    Properties p = new Properties();
		    try {

	p.load(classLoader.getResourceAsStream("i18n_"+idiomaActivo+".properties"));
				String valor = p.getProperty(texto);
				if (valor==null) {
					return "¿¿¿"+texto+"???";
				} else {
					return valor;
				}
			} catch (IOException e) {
				e.printStackTrace();
				return "¡¡¡Error!!!";
			}
		}
	}

	public static String getIdiomaActivo() {
		return idiomaActivo;
	}

	public static void setIdiomaActivo(String idiomaActivo) {
		LWFi18nService.idiomaActivo = idiomaActivo;
	}

	public static String[] getIdiomasAccesiblesTexto() {
		return idiomasAccesiblesTexto;
	}

	public static String[] getIdiomasAccesiblesCodigo() {
		return idiomasAccesiblesCodigo;
	}
}
```

7.1.1.5 LWFMappingFilter.java

```
package LWF.mapping;

import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Properties;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import LWF.util.LogUtil;
import LWF.util.PropertiesUtil;

public class LWFMappingFilter implements Filter
{
	@SuppressWarnings("rawtypes")
	@Override
	public void init(FilterConfig arg0) throws ServletException
```

```
	{
		LogUtil.print("LWFMappingFilter.init()");

		// Cargamos el fichero properties
		String path = arg0.getServletContext().getInitParameter("configuracio");
		Properties p = PropertiesUtil.load(path+"Mapping.properties");

		// Preparamos el hashmap para cargarlo en el contexto
		HashMap<String, String> hm = new HashMap<String, String>();

		// Recorremos todos sus elementos
		for (Enumeration e = p.keys(); e.hasMoreElements();) {
		    String key = (String) e.nextElement();
		    String value = p.getProperty(key);
		    LogUtil.print(key+" = "+value);
		    hm.put(key, value);
		}

		// Cargamos el hashmap para futuras consultas
		LWFMappingService.init(hm);
		LogUtil.print("Se ha cargado el mapeo de del path de la URL con sus respectivas
clases");
	}

	@SuppressWarnings({ "rawtypes", "unchecked" })
	@Override
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
	throws IOException, ServletException
	{
		String path = (String) request.getAttribute("LWF_PATH");
		String mapping = LWFMappingService.get(path);
		LogUtil.print("Path: "+path+" -> Maping: "+mapping);
	 try {
		// Clase
		Class c = Class.forName(mapping);
		Object o = c.newInstance();

		// Parametros del metodo
		Class[] p = new Class[2];
		p[0] = HttpServletRequest.class;
		p[1] = HttpServletResponse.class;

		// Metodo
		Method m = c.getMethod("userRequest",p);

		// Ahora los valores que pasaremos a los parametros
		HttpServletRequest pRequest = (HttpServletRequest) request;
		 HttpServletResponse pResponse = (HttpServletResponse) response;
		 Object[] input = {pRequest, pResponse};

		 // Lo invocamos
		 //m.invoke(new LoginUserRequest(), input);
		 m.invoke(o, input);

		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		} catch (InstantiationException e) {
			e.printStackTrace();
		} catch (IllegalAccessException e) {
			e.printStackTrace();
		} catch (IllegalArgumentException e) {
			e.printStackTrace();
		} catch (InvocationTargetException e) {
			e.printStackTrace();
		} catch (SecurityException e) {
			e.printStackTrace();
		} catch (NoSuchMethodException e) {
			e.printStackTrace();
		}
	}

	@Override
	public void destroy() {
```

```
                LogUtil.print("LWFMappingFilter.destroy()");
                LWFMappingService.destroy();
        }
}
```

7.1.1.6 LWFMappingService.java

```
package LWF.mapping;

import java.util.HashMap;

public class LWFMappingService
{
        private static HashMap<String, String> hmMapping = new HashMap<String, String>();

        public static void init(HashMap<String, String> hm)
        {
                hmMapping = hm;
        }

        public static String get(String key)
        {
                if (hmMapping.containsKey(key)) {
                        return hmMapping.get(key);
                } else {
                        return null;
                }
        }

        public static void destroy() {
                hmMapping = null;
        }
}
```

7.1.1.7 LWFMessageService.java

```
package LWF.message;

import javax.servlet.http.HttpServletRequest;

public class LWFMessageService
{
        private HttpServletRequest request = null;

        public LWFMessageService(HttpServletRequest req)
        {
                request = req;
        }

        public void addInfo(String valor)
        {
                request.setAttribute("MESSAGE", "INF_"+valor);
        }

        public void addError(String valor)
        {
                request.setAttribute("MESSAGE", "ERR_"+valor);
        }

        public static boolean existe(HttpServletRequest req)
        {
                String text = (String) req.getAttribute("MESSAGE");
                if (text==null) {
                        return false;
                } else {
                        return true;
                }
        }
```

```
	public static String printInHTML(HttpServletRequest req)
	{
		String text = (String) req.getAttribute("MESSAGE");
		if (text!=null) {
			if (text.startsWith("INF_")) {
				text = "<span style='color:blue'>"+text.substring(4)+"</span>";
			} else if (text.startsWith("ERR_")) {
				text = "<span style='color:red'>"+text.substring(4)+"</span>";
			} else {
				text = "<span style='color:grey'>"+text+"</span>";
			}
		}
		return text;
	}
}
```

7.1.1.8 LWFUserRequestInterface.java

```
package LWF.userRequest;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public interface LWFUserRequestInterface
{
	public void userRequest(HttpServletRequest req, HttpServletResponse resp) throws
Exception;
}
```

7.1.1.9 LogUtil.java

```
package LWF.util;

public class LogUtil
{
	public static void print(String texto)
	{
		System.out.println(">>> "+texto);
	}
}
```

7.1.1.10 PropertiesUtil.java

```
package LWF.util;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class PropertiesUtil
{
	public static Properties load(String fileName)
	{
		Properties p = new Properties();
		InputStream is = null;
		File f = null;
		try {
			f = new File(fileName);
			is = new FileInputStream(f);
			p.load(is);
		} catch(IOException ioe) {
			p=null;
		}
		return p;
```

```
	}
}
```

7.1.1.11 SAXUtil.java

```
package LWF.util;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

public class SAXUtil
{
	public static Document load(String fileName)
	{
		// Creamos el builder basado en SAX
		SAXBuilder builder = new SAXBuilder();
		// Construimos el arbol DOM a partir del fichero xml
		Document documentJDOM = null;
		try {
			documentJDOM = builder.build(new FileInputStream(fileName));
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (JDOMException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
		return documentJDOM;
	}
}
```

7.1.1.12 LWFValidationFilter.java

```
package LWF.validation;

import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import org.jdom.Document;
import org.jdom.Element;

import LWF.i18n.LWFi18nService;
import LWF.message.LWFMessageService;
import LWF.util.LogUtil;
import LWF.util.SAXUtil;

public class LWFValidationFilter implements Filter
{
	@SuppressWarnings("unchecked")
	@Override
	public void init(FilterConfig arg0)
	throws ServletException
```

```
	{
		LogUtil.print("LWFValidationFilter.init()");

		// Cargamos el fichero
	String path = arg0.getServletContext().getInitParameter("configuracio");
		Document doc = SAXUtil.load(path+"Validation.xml");

		// Obtenemos la etiqueta raíz
		Element n1 = doc.getRootElement();

		// Recorremos los hijos de la etiqueta raíz
		List<Element> n1Hijos = n1.getChildren();
		HashMap<String, HashMap<String, String>> hmFormularios = new HashMap<String,
HashMap<String, String>>();
		for (Element n2: n1Hijos) {

			// Obtenemos el nombre y su contenido de tipo texto
			String n2Elemento = n2.getName();
			String n2Nombre = n2.getAttributeValue("name");
			LogUtil.print("Elemento: "+n2Elemento+", name: "+n2Nombre);

			List<Element> n2Hijos = n2.getChildren();
			HashMap<String, String> hmAtributos = new HashMap<String, String>();
			for (Element n3: n2Hijos) {

				// Obtenemos el nombre y su contenido de tipo texto
				String n3Elemento = n3.getName();
				String n3Nombre = n3.getAttributeValue("name");
				String n3Metodos = n3.getAttributeValue("method");
				LogUtil.print("Elemento: "+n3Elemento+", name: "+n3Nombre+",
method: "+n3Metodos);
				hmAtributos.put(n3Nombre, n3Metodos);
			}
			hmFormularios.put(n2Nombre, hmAtributos);
		}
		LWFValidationService.init(hmFormularios);
	}

	@SuppressWarnings({ "unchecked", "rawtypes" })
	@Override
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
	throws IOException, ServletException
	{
		Object oForm = request.getAttribute("LWF_FORM");
		if (oForm!=null) {

			Class cForm = oForm.getClass();
			String sFormClassName = cForm.getSimpleName();

			HashMap<String, String> hmValidaciones =
LWFValidationService.get(sFormClassName);
			LogUtil.print("formulario: "+sFormClassName+" -> validaciones:
"+hmValidaciones);
		 try {
			// Recuperamos los parametros a validar
			List<String> lValidaciones = new
ArrayList<String>(hmValidaciones.keySet());

			// Clases
			//Class cForm = Class.forName(sFormCompleteName);
			//Object oForm = cForm.newInstance();
			HttpServletRequest userRequest = (HttpServletRequest) request;
			String methodsClass =
userRequest.getSession().getServletContext().getInitParameter("validationMethodsClass");
			Class cVal = Class.forName(methodsClass);
			Object oVal = cVal.newInstance();

			// Parametros de los metodos
			Class[] p = new Class[1];
			p[0] = String.class;

			// Parametros del validador
			for (String parametro: lValidaciones) {
```

```
                                String sVal = hmValidaciones.get(parametro);
                                String[] aVal = null;
                                if (sVal!=null && sVal.contains(",")) {
                                        aVal = sVal.split(",");
                                } else if (sVal!=null) {
                                        aVal = new String[1];
                                        aVal[0] = sVal;
                                }

                                // Aplicamos las validaciones
                                for (int i=0; i<aVal.length; i++) {
                                        // Metodos
                                        String metodoForm = "get"+parametro.substring(0,
1).toUpperCase()+parametro.substring(1);
                                        Method mForm = cForm.getMethod(metodoForm);
                                Method mVal = cVal.getMethod(aVal[i],p);
                                // Ahora los valores que pasaremos a los parametros
                                Object[] aObj = null;
                                 String valor = (String) mForm.invoke(oForm, aObj);
                                 Object[] input = {valor};

                                 // Lo invocamos
                                 Boolean output = (Boolean) mVal.invoke(oVal, input);
                                 LogUtil.print(parametro+", "+aVal[i]+", "+output.toString());
                                 if (output.booleanValue()!=true) {
                                                LWFMessageService mess = new
LWFMessageService((HttpServletRequest) request);

        mess.addError(LWFi18nService.traduce("missatge.error.validacio")+": "+parametro+",
"+aVal[i]+", "+output.toString());
                                        chain.doFilter(request, response);
                                        return;
                                 }
                                }
                        }
                        } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                        } catch (InstantiationException e) {
                                e.printStackTrace();
                        } catch (IllegalAccessException e) {
                                e.printStackTrace();
                        } catch (IllegalArgumentException e) {
                                e.printStackTrace();
                        } catch (InvocationTargetException e) {
                                e.printStackTrace();
                        } catch (SecurityException e) {
                                e.printStackTrace();
                        } catch (NoSuchMethodException e) {
                                e.printStackTrace();
                        }
                }
                chain.doFilter(request, response);
        }

        @Override
        public void destroy() {
                LogUtil.print("LWFValidationFilter.destroy()");
                LWFValidationService.destroy();
        }
}
```

7.1.1.13 LWFValidationMethods.java

```
package LWF.validation;

public class LWFValidationMethods
{
        public Boolean isInteger(String value)
        {
                try {
                        Integer.parseInt(value);
```

```
                return new Boolean("true");
            } catch(Exception e) {
                return new Boolean("false");
            }
        }

        public Boolean isPasswordPattern(String value)
        {
            try {
                if (value.length()<8) return new Boolean("false");
                if (!value.matches(".*[A-Z].*")) return new Boolean("false");
                if (!value.matches(".*[a-z].*")) return new Boolean("false");
                if (!value.matches(".*[0-9].*")) return new Boolean("false");
                return new Boolean("true");
            } catch(Exception e) {
                return new Boolean("false");
            }

        }
}
```

7.1.1.14 LWFValidationService.java

```
package LWF.validation;

import java.util.HashMap;

public class LWFValidationService
{
        private static HashMap<String, HashMap<String, String>> hmForm = new HashMap<String,
HashMap<String, String>>();

        public static void init(HashMap<String, HashMap<String, String>> hm)
        {
            hmForm = hm;
        }

        public static HashMap<String, String> get(String key)
        {
            if (hmForm.containsKey(key)) {
                return hmForm.get(key);
            } else {
                return null;
            }
        }

        public static void destroy() {
            hmForm = null;
        }
}
```

7.1.1.15 LWFViewFilter.java

```
package LWF.view;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import org.jdom.Document;
```

```
import org.jdom.Element;

import LWF.util.LogUtil;
import LWF.util.SAXUtil;

public class LWFViewFilter implements Filter
{
	@SuppressWarnings("unchecked")
	@Override
	public void init(FilterConfig arg0) throws ServletException
	{
		LogUtil.print("LWFViewFilter.init()");

		// Cargamos el fichero
	String path = arg0.getServletContext().getInitParameter("configuracio");
		Document doc = SAXUtil.load(path+"View.xml");

		// Obtenemos la etiqueta raíz
		Element n1 = doc.getRootElement();

		// Recorremos los hijos de la etiqueta raíz
		List<Element> n1Hijos = n1.getChildren();
		HashMap<String, HashMap<String, String>> hmFormularios = new HashMap<String,
HashMap<String, String>>();
		for (Element n2: n1Hijos) {

			// Obtenemos el nombre y su contenido de tipo texto
			String n2Elemento = n2.getName();
			String n2Nombre = n2.getAttributeValue("name");
			String n2Plantilla = n2.getAttributeValue("template");
			LogUtil.print("Elemento: "+n2Elemento+", name: "+n2Nombre);
			LogUtil.print("Elemento: "+n2Elemento+", templte: "+n2Plantilla);

			List<Element> n2Hijos = n2.getChildren();
			HashMap<String, String> hmAtributos = new HashMap<String, String>();
			hmAtributos.put("template", n2Plantilla);

			for (Element n3: n2Hijos) {

				// Obtenemos el nombre y su contenido de tipo texto
				String n3Elemento = n3.getName();
				String n3Nombre = n3.getAttributeValue("name");
				String n3Valor = n3.getAttributeValue("value");
				LogUtil.print("Elemento: "+n3Elemento+", name: "+n3Nombre+",
value: "+n3Valor);
				hmAtributos.put(n3Nombre, n3Valor);
			}
			hmFormularios.put(n2Nombre, hmAtributos);
		}
		LWFViewService.init(hmFormularios);
	}

	@Override
	public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2)
	throws IOException, ServletException {
		LogUtil.print("LWFViewFilter.doFilter()");
	}

	@Override
	public void destroy() {
		LogUtil.print("LWFViewFilter.destroy()");
		LWFViewService.destroy();
	}

	public static String map(HttpServletRequest req, String name)
	throws IOException, ServletException
	{
		LogUtil.print("LWFViewFilter.map()");

		// Leemos la configuración a partir del nombre
		HashMap<String, String> hmView = LWFViewService.get(name);
		LogUtil.print("view: "+name+" -> valores: "+hmView);

		// Recuperamos las partes de la plantilla y las subimos a sessión
```

```
                List<String> lView = new ArrayList<String>(hmView.keySet());
                for (int i=0; i<lView.size(); i++) {
                        String elemento = lView.get(i);
                        String valor = hmView.get(elemento);
                        req.setAttribute("TEMPLATE_"+elemento, valor);
                        LogUtil.print("Subido al request: TEMPLATE_"+elemento+" -> valor:
"+valor);
                }

                // Devolvemos la plantilla
                return hmView.get("template");
        }
}
```

7.1.1.16 LWFViewService.java

```
package LWF.view;

import java.util.HashMap;

public class LWFViewService
{
        private static HashMap<String, HashMap<String, String>> hmView = new HashMap<String,
HashMap<String, String>>();

        public static void init(HashMap<String, HashMap<String, String>> hm)
        {
                hmView = hm;
        }

        public static HashMap<String, String> get(String key)
        {
                if (hmView.containsKey(key)) {
                        return hmView.get(key);
                } else {
                        return null;
                }
        }

        public static void destroy() {
                hmView = null;
        }
}
```

1.1.2 Fitxers de configuració

7.1.2.1 i18n_CA.properties

```
missatge.login.error=La autentificació ha fallat! torna-ho a intentar...
missatge.login.ok=La autentificació ha sigut correcta!
missatge.error.validacio=Error en la validació
```

7.1.2.1 Mapping.properties

```
# http:\\www.dominio.com\contexto\path
# Aqui sólo se mapea el path con la clase que implementa un UserRequestInterface

login = com.aplicacion.peticiones.LoginUserRequest
autentificacion = com.aplicacion.peticiones.LoginUserRequest
home = com.aplicacion.peticiones.HomeUserRequest
```

7.1.2.2 Validation.xml

```
<validation>
	<form name="LoginUserForm">
		<input name="user" method="isInteger"/>
		<input name="password" method="isInteger,isPasswordPattern"/>
	</form>
</validation>
```

7.1.2.3 View.xml

```
<!--
Pàgina=pantalla.manteniment.usuari
Plantilla=pantalla_manteniment.jsp
Element.cap=capçalera.jsp
Element.ms=menusuperior.jsp
Element.me=menuesquerra.jsp
Element.cos=manteniment_usuari.jsp
Element.peu=peu.jsp
-->
<view>
	<page name="pantalla.manteniment.usuari" template="pantalla_manteniment">
		<element name="cap" value="capçalera.jsp"/>
		<element name="ms" value="menusuperior.jsp"/>
		<element name="me" value="menuesquerra.jsp"/>
		<element name="cos" value="manteniment_usuari.jsp"/>
		<element name="peu" value="peu.jsp"/>
	</page>
</view>
```

7.1.2.3 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app	id="WebApp_ID"	version="2.4"	xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee	http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">

	<display-name>LWF</display-name>

	<context-param>
		<param-name>validationMethodsClass</param-name>
		<param-value>LWF.validation.LWFValidationMethods</param-value>
	</context-param>

	<filter>
		<filter-name>controller</filter-name>
		<filter-class>LWF.controller.LWFControllerFilter</filter-class>
	</filter>

	<filter>
		<filter-name>formExtractor</filter-name>
		<filter-class>LWF.formExtractor.LWFFormExtractorFilter</filter-class>
	</filter>

	<filter>
		<filter-name>validation</filter-name>
		<filter-class>LWF.validation.LWFValidationFilter</filter-class>
	</filter>

	<filter>
		<filter-name>mapping</filter-name>
		<filter-class>LWF.mapping.LWFMappingFilter</filter-class>
	</filter>

	<filter-mapping>
		<filter-name>controller</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>

	<filter-mapping>
```

```
		<filter-name>formExtractor</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>

	<filter-mapping>
		<filter-name>validation</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>

	<filter-mapping>
		<filter-name>mapping</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>

</web-app>
```

7.1.2.4 build.xml

```
<?xml version="1.0"?>
<project name="LWF" default="jar" basedir=".">

	<!-- ========== -->
	<!-- PROPERTIES -->
	<!-- ========== -->

	<property name="APP" value="LWF" />
	<property name="JAR" value="${APP}.jar" />
	<property name="SRC" value="src" />
	<property name="CLASSES" value="WebContent/WEB-INF/classes" />
	<property name="WEB" value="WebContent" />
	<property name="DEPLOY" value="dist" />
	<property name="DIRECTORIO" value="../UOC_PFC_PAC4/WebContent/WEB-INF/lib" />

	<!-- === -->
	<!-- JAR -->
    <!-- === -->

	<target name="jar">
		<delete file="${DEPLOY}/${JAR}"/>
		<jar destfile="${DEPLOY}/${JAR}">
			<fileset dir="${CLASSES}">
				<exclude name="*.properties"/>
			 <exclude name="*.xml"/>
			</fileset>
		</jar>
		<delete file="${DIRECTORIO}/${JAR}"/>
		<copy file="${DEPLOY}/${JAR}" tofile="${DIRECTORIO}/${JAR}" overwrite="true"/>
	</target>

</project>
```

1.2 Directori

1.2.1 Classes

7.2.1.1 AssociarUsuarisGrupsUserForm.java

```
package com.directori.formularis;

public class AssociarUsuarisGrupsUserForm
{
	private String idUsuari = null;
	private String idGrup = null;

	public String getIdUsuari() {
		return idUsuari;
	}
```

```
	public void setIdUsuari(String idUsuari) {
		this.idUsuari = idUsuari;
	}
	public String getIdGrup() {
		return idGrup;
	}
	public void setIdGrup(String idGrup) {
		this.idGrup = idGrup;
	}
}
```

7.2.1.2 AutenticarUserForm.java

```
package com.directori.formularis;

public class AutenticarUserForm
{
	private String user = null;
	private String password = null;

	public String getUser() {
		return user;
	}

	public void setUser(String user) {
		this.user = user;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}
}
```

7.2.1.3 GrupsUserForm.java

```
package com.directori.formularis;

public class GrupsUserForm
{
	private String id = null;
	private String nom = null;
	private String descripcio = null;

	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getNom() {
		return nom;
	}
	public void setNom(String nom) {
		this.nom = nom;
	}
	public String getDescripcio() {
		return descripcio;
	}
	public void setDescripcio(String descripcio) {
		this.descripcio = descripcio;
	}
}
```

7.2.1.4 UsuarisUserForm.java

```
package com.directori.formularis;

public class UsuarisUserForm
{
	private String id = null;
	private String nom = null;
	private String cognom1 = null;
	private String cognom2 = null;
	private String mobil = null;
	private String mail = null;

	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getNom() {
		return nom;
	}
	public void setNom(String nom) {
		this.nom = nom;
	}
	public String getCognom1() {
		return cognom1;
	}
	public void setCognom1(String cognom1) {
		this.cognom1 = cognom1;
	}
	public String getCognom2() {
		return cognom2;
	}
	public void setCognom2(String cognom2) {
		this.cognom2 = cognom2;
	}
	public String getMobil() {
		return mobil;
	}
	public void setMobil(String mobil) {
		this.mobil = mobil;
	}
	public String getMail() {
		return mail;
	}
	public void setMail(String mail) {
		this.mail = mail;
	}
}
```

7.2.1.5 AssociarUsuarisGrupsPersistencia.java

```
package com.directori.persistencia;

import java.util.ArrayList;

import com.directori.formularis.AssociarUsuarisGrupsUserForm;

public class AssociarUsuarisGrupsPersistencia
{
	private static ArrayList<AssociarUsuarisGrupsUserForm> llistaAssociasion = new
ArrayList<AssociarUsuarisGrupsUserForm>();

	public static void guardar(AssociarUsuarisGrupsUserForm usuari) {
		llistaAssociasion.add(usuari);
	}

	public static ArrayList<AssociarUsuarisGrupsUserForm> recuperar() {
		return llistaAssociasion;
```

```
	}
}
```

7.2.1.6 GrupsPersistencia.java

```
package com.directori.persistencia;

import java.util.ArrayList;

import com.directori.formularis.GrupsUserForm;

public class GrupsPersistencia
{
	private static ArrayList<GrupsUserForm> llistaGrups = new ArrayList<GrupsUserForm>();

	public static void guardar(GrupsUserForm grup) {
		llistaGrups.add(grup);
	}

	public static ArrayList<GrupsUserForm> recuperar() {
		return llistaGrups;
	}
}
```

7.2.1.7 UsuarisPersistencia.java

```
package com.directori.persistencia;

import java.util.ArrayList;

import com.directori.formularis.UsuarisUserForm;

public class UsuarisPersistencia
{
	private static ArrayList<UsuarisUserForm> llistaUsuaris = new
ArrayList<UsuarisUserForm>();

	public static void guardar(UsuarisUserForm usuari) {
		llistaUsuaris.add(usuari);
	}

	public static ArrayList<UsuarisUserForm> recuperar() {
		return llistaUsuaris;
	}
}
```

7.2.1.8 AssociarUsuarisGrupsUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import LWF.message.LWFMessageService;
import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;
import LWF.view.LWFViewFilter;

import com.directori.formularis.AssociarUsuarisGrupsUserForm;
import com.directori.persistencia.AssociarUsuarisGrupsPersistencia;
import com.directori.persistencia.GrupsPersistencia;
import com.directori.persistencia.UsuarisPersistencia;

public class AssociarUsuarisGrupsUserRequest implements LWFUserRequestInterface
{
```

```
	@Override
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("AssociarUsuarisGrupsUserRequest.userRequest()");

		AssociarUsuarisGrupsUserForm form = (AssociarUsuarisGrupsUserForm)
req.getAttribute("LWF_FORM");
		if (form!=null) {

			if (!LWFMessageService.existe(req)) {
				// Cream
				LogUtil.print(form.getIdUsuari());
				LogUtil.print(form.getIdGrup());
				AssociarUsuarisGrupsPersistencia.guardar(form);
			}
		}

		// Sessió
		req.setAttribute("LLISTAT", AssociarUsuarisGrupsPersistencia.recuperar());
		req.setAttribute("LLISTAT_USUARIS", UsuarisPersistencia.recuperar());
		req.setAttribute("LLISTAT_GRUPS", GrupsPersistencia.recuperar());

		// Redirigim
		RequestDispatcher dispatcher = req.getRequestDispatcher(LWFViewFilter.map(req,
"pantalla.manteniment.associar"));
		dispatcher.forward(req, res);
	}
}
```

7.2.1.9 AutenticarUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.directori.formularis.AutenticarUserForm;

import LWF.i18n.LWFi18nService;
import LWF.message.LWFMessageService;
import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;
import LWF.view.LWFViewFilter;

public class AutenticarUserRequest implements LWFUserRequestInterface
{
	@Override
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("AutenticarUserRequest.userRequest()");

		try {
			AutenticarUserForm form = (AutenticarUserForm)
req.getAttribute("LWF_FORM");

			String user = form.getUser();
			String pass = form.getPassword();
			LogUtil.print(user);
			LogUtil.print(pass);
			if (!"admin".equals(user) || !"1234".equals(pass)) throw new Exception("
Usuario o password incorrectos");

			LWFMessageService mess = new LWFMessageService(req);
			mess.addInfo(LWFi18nService.traduce("missatge.login.ok"));
			RequestDispatcher dispatcher =
req.getRequestDispatcher(LWFViewFilter.map(req, "pantalla.manteniment.home"));
			dispatcher.forward(req, res);
```

```
		} catch(Exception e) {
			LWFMessageService mess = new LWFMessageService(req);
			mess.addError(LWFi18nService.traduce("missatge.login.error"));
			RequestDispatcher dispatcher =
req.getRequestDispatcher(LWFViewFilter.map(req, "pantalla.login"));
			dispatcher.forward(req, res);
		}
	}
}
```

7.2.1.10 GrupsUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import LWF.message.LWFMessageService;
import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;
import LWF.view.LWFViewFilter;

import com.directori.formularis.GrupsUserForm;
import com.directori.persistencia.GrupsPersistencia;

public class GrupsUserRequest implements LWFUserRequestInterface
{
	@Override
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("GrupsUserRequest.userRequest()");

		GrupsUserForm form = (GrupsUserForm) req.getAttribute("LWF_FORM");
		if (form!=null) {

			if (!LWFMessageService.existe(req)) {
				// Cream
				LogUtil.print(form.getId());
				LogUtil.print(form.getNom());
				LogUtil.print(form.getDescripcio());
				GrupsPersistencia.guardar(form);
			}
		}
		// Sessió
		req.setAttribute("LLISTAT", GrupsPersistencia.recuperar());

		// Redirigim
		RequestDispatcher dispatcher = req.getRequestDispatcher(LWFViewFilter.map(req,
"pantalla.manteniment.grups"));
		dispatcher.forward(req, res);
	}
}
```

7.2.1.11 HomeUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;

public class HomeUserRequest implements LWFUserRequestInterface
{
	@Override
```

```
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("HomeUserRequest.userRequest()");

		RequestDispatcher dispatcher = req.getRequestDispatcher("peticions/home.jsp");
		dispatcher.forward(req, res);
	}
}
```

7.2.1.12 LoginUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;
import LWF.view.LWFViewFilter;

public class LoginUserRequest implements LWFUserRequestInterface
{
	@Override
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("LoginUserRequest.userRequest()");

		RequestDispatcher dispatcher = req.getRequestDispatcher(LWFViewFilter.map(req,
"pantalla.login"));
		LogUtil.print("LoginUserRequest.userRequest() - dispatcher");
		dispatcher.forward(req, res);
	}
}
```

7.2.1.13 UsuarisUserRequest.java

```
package com.directori.peticions;

import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.directori.formularis.UsuarisUserForm;
import com.directori.persistencia.UsuarisPersistencia;

import LWF.i18n.LWFi18nService;
import LWF.message.LWFMessageService;
import LWF.userRequest.LWFUserRequestInterface;
import LWF.util.LogUtil;
import LWF.view.LWFViewFilter;

public class UsuarisUserRequest implements LWFUserRequestInterface
{
	@Override
	public void userRequest(HttpServletRequest req, HttpServletResponse res)
	throws Exception
	{
		LogUtil.print("UsuarisUserRequest.userRequest()");

		UsuarisUserForm form = (UsuarisUserForm) req.getAttribute("LWF_FORM");
		if (form!=null) {

			if (!LWFMessageService.existe(req)) {
				// Cream
				LogUtil.print(form.getId());
				LogUtil.print(form.getNom());
```

```
                                LogUtil.print(form.getCognom1());
                                LogUtil.print(form.getCognom2());
                                LogUtil.print(form.getMobil());
                                LogUtil.print(form.getMail());
                                UsuarisPersistencia.guardar(form);
                        }
                }

                // Sessió
                req.setAttribute("LLISTAT", UsuarisPersistencia.recuperar());

                // Redirigim
                RequestDispatcher dispatcher = req.getRequestDispatcher(LWFViewFilter.map(req,
"pantalla.manteniment.usuaris"));
                dispatcher.forward(req, res);
        }
}
```

7.2.1.14 Validacions.java

```
package com.directori.validacions;

import LWF.validation.LWFValidationMethods;

public class Validacions extends LWFValidationMethods
{
        public Boolean isMandatory(String text)
        {
                if (text==null || "".equals(text)) {
                        return new Boolean(false);
                } else {
                        return new Boolean(true);
                }
        }

        public Boolean isString(String text)
        {
                if (text.matches("[a-z|A-Z]+")) {
                        return new Boolean(true);
                } else {
                        return new Boolean(false);
                }
        }

        public Boolean isMobil(String text)
        {
                if (text.matches("6\\d{8}")) {
                        return new Boolean(true);
                } else {
                        return new Boolean(false);
                }
        }

        public Boolean isMail(String text)
        {
                if (text.matches("\\w+@\\w+\\.\\w+")) {
                        return new Boolean(true);
                } else {
                        return new Boolean(false);
                }
        }
}
```

1.2.2 Fitxers de configuració

7.2.2.1 i18n_CA.properties

```
diccionari.formulari=Formulari
```

```
diccionari.llistat=LListat

missatge.login.error=La autentificació ha fallat! torna-ho a intentar...
missatge.login.ok=La autentificació ha sigut correcta!
missatge.error.validacio=Error en la validació

login.titulo=Login
login.usuari=Usuari
login.password=Clau
login.submit=Accedeix

manteniment.titol=Directori
manteniment.logout=Sortir
manteniment.peu=Melchor Vives Bernat - PFC - Enginyeria en Informàtica - UOC
manteniment.missatge=Missatge

manteniment.menu.titol=Menú
manteniment.menu.usuaris=Gestió d'usuaris
manteniment.menu.grups=Gestió de grups
manteniment.menu.associar=Associar usuaris amb grups

manteniment.usuari.id=Id
manteniment.usuari.nom=Nom
manteniment.usuari.cognom1=Cognom1
manteniment.usuari.cognom2=Cognom2
manteniment.usuari.mobil=Mòbil
manteniment.usuari.mail=Mail
manteniment.usuari.submit=Crear

manteniment.grup.id=Id
manteniment.grup.nom=Nom
manteniment.grup.descripcio=Descripció
manteniment.grup.submit=Crear

manteniment.associar.usuari=Id usuari
manteniment.associar.grup=Id grup
manteniment.associar.submit=Associar
```

7.2.2.2 i18n_EN.properties

```
diccionari.formulari=Form
diccionari.llistat=List

missatge.login.error=Authentication failed! Try again ...
missatge.login.ok=Authentication was successful!
missatge.error.validacio=Failed validation

login.titulo=Login
login.usuari=User
login.password=Password
login.submit=Sign in

manteniment.titol=Directory
manteniment.logout=Logout
manteniment.peu=Melchor Vives Bernat - PFC - Ingeniería en Informática - UOC
manteniment.missatge=Message

manteniment.menu.titol=Menú
manteniment.menu.usuaris=User management
manteniment.menu.grups=Group management
manteniment.menu.associar=Associate users with groups

manteniment.usuari.id=Id
manteniment.usuari.nom=Name
manteniment.usuari.cognom1=Surname1
manteniment.usuari.cognom2=Surname2
manteniment.usuari.mobil=Mobile
manteniment.usuari.mail=Mail
manteniment.usuari.submit=Create

manteniment.grup.id=Id
```

```
manteniment.grup.nom=Name
manteniment.grup.descripcio=Descriptión
manteniment.grup.submit=Create

manteniment.associar.usuari=Id user
manteniment.associar.grup=Id group
manteniment.associar.submit=Associate
```

7.2.2.3 i18n_ES.properties

```
diccionari.formulari=Formulario
diccionari.llistat=LListado

missatge.login.error=La autentificación ha fallado! vuelvelo a intentar...
missatge.login.ok=La autentificación ha sido correcta!
missatge.error.validacio=Error en la validación

login.titulo=Login
login.usuari=Usuario
login.password=Contraseña
login.submit=Accede

manteniment.titol=Directorio
manteniment.logout=Salir
manteniment.peu=Melchor Vives Bernat - PFC - Ingeniería en Informática - UOC
manteniment.missatge=Mensaje

manteniment.menu.titol=Menú
manteniment.menu.usuaris=Gestión de usuarios
manteniment.menu.grups=Gestión de grupos
manteniment.menu.associar=Asociar usuarios con grupos

manteniment.usuari.id=Id
manteniment.usuari.nom=Nombre
manteniment.usuari.cognom1=Apellido1
manteniment.usuari.cognom2=Apellido2
manteniment.usuari.mobil=Móvil
manteniment.usuari.mail=Mail
manteniment.usuari.submit=Crear

manteniment.grup.id=Id
manteniment.grup.nom=Nom
manteniment.grup.descripcio=Descripción
manteniment.grup.submit=Crear

manteniment.associar.usuari=Id usuario
manteniment.associar.grup=Id grupo
manteniment.associar.submit=Asociar
```

7.2.2.4 Mapping.properties

```
# http:\\www.dominio.com\contexto\path
# Aqui sólo se mapea el path con la clase que implementa un UserRequestInterface

login = com.directori.peticions.LoginUserRequest
autenticar = com.directori.peticions.AutenticarUserRequest
home = com.directori.peticions.HomeUserRequest
usuaris = com.directori.peticions.UsuarisUserRequest
grups = com.directori.peticions.GrupsUserRequest
associar-usuaris-grups = com.directori.peticions.AssociarUsuarisGrupsUserRequest
```

7.2.2.5 Validation.xml

```
<validation>
       <form name="AutenticarUserForm">
              <input name="user" method="isMandatory,isString"/>
```

```
            <input name="password" method="isMandatory,isPasswordPattern"/>
        </form>
        <form name="UsuarisUserForm">
            <input name="id" method="isMandatory,isInteger"/>
            <input name="nom" method="isMandatory,isString"/>
        </form>
        <form name="GrupsUserForm">
            <input name="id" method="isMandatory,isInteger"/>
            <input name="nom" method="isMandatory,isString"/>
        </form>
        <form name="AssociarUsuarisGrupsUserForm">
            <input name="idUsuari" method="isMandatory,isInteger"/>
            <input name="idGrup" method="isMandatory,isInteger"/>
        </form>
</validation>
```

7.2.2.6 View.xml

```
<view>
        <page name="pantalla.login" template="plantilles/login.jsp">
            <element name="missatge" value="/generic/missatge.jsp"/>
            <element name="cos" value="/pantalles/login/cos.jsp"/>
        </page>
        <page name="pantalla.manteniment.home" template="plantilles/manteniment.jsp">
            <element name="cap" value="/generic/cap.jsp"/>
            <element name="missatge" value="/generic/missatge.jsp"/>
            <element name="menu" value="/generic/menu.jsp"/>
            <element name="cos" value=""/>
            <element name="llistat" value=""/>
            <element name="peu" value="/generic/peu.jsp"/>
        </page>
        <page name="pantalla.manteniment.usuaris" template="plantilles/manteniment.jsp">
            <element name="cap" value="/generic/cap.jsp"/>
            <element name="missatge" value="/generic/missatge.jsp"/>
            <element name="menu" value="/generic/menu.jsp"/>
            <element name="cos" value="/pantalles/usuaris/cos.jsp"/>
            <element name="llistat" value="/pantalles/usuaris/llistat.jsp"/>
            <element name="peu" value="/generic/peu.jsp"/>
        </page>
        <page name="pantalla.manteniment.grups" template="plantilles/manteniment.jsp">
            <element name="cap" value="/generic/cap.jsp"/>
            <element name="missatge" value="/generic/missatge.jsp"/>
            <element name="menu" value="/generic/menu.jsp"/>
            <element name="cos" value="/pantalles/grups/cos.jsp"/>
            <element name="llistat" value="/pantalles/grups/llistat.jsp"/>
            <element name="peu" value="/generic/peu.jsp"/>
        </page>
        <page name="pantalla.manteniment.associar" template="plantilles/manteniment.jsp">
            <element name="cap" value="/generic/cap.jsp"/>
            <element name="missatge" value="/generic/missatge.jsp"/>
            <element name="menu" value="/generic/menu.jsp"/>
            <element name="cos" value="/pantalles/associar/cos.jsp"/>
            <element name="llistat" value="/pantalles/associar/llistat.jsp"/>
            <element name="peu" value="/generic/peu.jsp"/>
        </page>
</view>
```

7.2.2.7 build.xml

```
<?xml version="1.0"?>
<project name="Directori" default="war" basedir=".">

        <!-- ========== -->
        <!-- PROPERTIES -->
        <!-- ========== -->

        <property name="APP" value="Directori" />
        <property name="WAR" value="${APP}.war" />
        <property name="SRC" value="src" />
```

```
        <property name="CLASSES" value="WebContent/WEB-INF/classes" />
        <property name="WEB" value="WebContent" />
        <property name="DEPLOY" value="dist" />

        <!-- === -->
        <!-- WAR -->
    <!-- === -->

        <target name="war">
                <delete file="${DEPLOY}/${WAR}"/>
                <war destfile="${DEPLOY}/${WAR}" webxml="${WEB}/WEB-INF/web.xml">
                        <fileset dir="${WEB}">
                                <include name="**" />
                        </fileset>
                </war>
        </target>

</project>
```

1.2.3 JSPs

7.2.3.1 directori/generic/cap.jsp

```
<%@page import="LWF.i18n.LWFi18nService" %>

<div style="width:100%; text-
align:center;"><h2><%=LWFi18nService.traduce("manteniment.titol")%></h2></div>
```

7.2.3.2 directori/generic/menu.jsp

```
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Menú superior -->
<fieldset>
        <legend><%=LWFi18nService.traduce("manteniment.menu.titol")%></legend>
        <a
href="/directori/usuaris"><%=LWFi18nService.traduce("manteniment.menu.usuaris")%></a>
         - <a
href="/directori/grups"><%=LWFi18nService.traduce("manteniment.menu.grups")%></a>
         - <a href="/directori/associar-usuaris-
grups"><%=LWFi18nService.traduce("manteniment.menu.associar")%></a>
</fieldset>
<br/>
```

7.2.3.3 directori/generic/missatge.jsp

```
<%@page import="LWF.message.LWFMessageService" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<%
if (LWFMessageService.existe(request)) {
%>
<br/>
<!-- Message -->
<fieldset>
        <legend><%=LWFi18nService.traduce("manteniment.missatge")%></legend>
        <%= LWFMessageService.printInHTML(request) %><br/>
</fieldset>
<%
}
%>
<br/>
```

7.2.3.4 directori/generic/peu.jsp

```
<%@page import="LWF.i18n.LWFi18nService" %>

<div style="width:100%; text-align:center;">
	<i><%=LWFi18nService.traduce("manteniment.peu")%></i>
</div>
```

7.2.3.5 directori/pantalles/associar/cos.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.AssociarUsuarisGrupsUserForm" %>
<%@page import="com.directori.formularis.UsuarisUserForm" %>
<%@page import="com.directori.formularis.GrupsUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Formulari -->
<fieldset>
	<legend><%=LWFi18nService.traduce("diccionari.formulari")%></legend>
	<form method="POST" action="/directori/associar-usuaris-grups">
		<input type="hidden" id="form" name="form"
value="com.directori.formularis.AssociarUsuarisGrupsUserForm"/><br/>
		<table>
			<tr>

	<td><%=LWFi18nService.traduce("manteniment.associar.usuari")%></td>
				<td>
					<select id="idUsuari" name="idUsuari"
style="width:200px">
						<option selected="selected"></option>
						<%
						ArrayList<UsuarisUserForm> llistatUsuaris =
(ArrayList<UsuarisUserForm>) request.getAttribute("LLISTAT_USUARIS");
						if (llistatUsuaris!=null) {
							for (int i=0; i<llistatUsuaris.size();
i++) {
								UsuarisUserForm usu =
llistatUsuaris.get(i);
						%>
						<option value="<%=usu.getId()%>"><%=usu.getId()%>
- <%=usu.getNom()%></option>
						<%
							}
						}
						%>
					</select>
				</td>
			</tr>
			<tr>

	<td><%=LWFi18nService.traduce("manteniment.associar.grup")%></td>
				<td>
					<select id="idGrup" name="idGrup" style="width:200px">
						<option selected="selected"></option>
						<%
						ArrayList<GrupsUserForm> llistatGrups =
(ArrayList<GrupsUserForm>) request.getAttribute("LLISTAT_GRUPS");
						if (llistatGrups!=null) {
							for (int i=0; i<llistatGrups.size(); i++)
{
								GrupsUserForm bean =
llistatGrups.get(i);
						%>
						<option
value="<%=bean.getId()%>"><%=bean.getId()%> - <%=bean.getNom()%></option>
						<%
							}
						}
						%>
```

```
                                        </select>
                                </td>
                        </tr>
                        <tr>
                                <td colspan="2" style="text-align:right;"><input type="submit"
value="<%=LWFi18nService.traduce("manteniment.associar.submit")%>"/></td>
                        </tr>
                </table>
        </form>
</fieldset>
<br/>
```

7.2.3.6 directori/pantalles/associar/llistat.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.AssociarUsuarisGrupsUserForm" %>
<%@page import="com.directori.formularis.UsuarisUserForm" %>
<%@page import="com.directori.formularis.GrupsUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Llistat -->
<fieldset>
        <legend><%=LWFi18nService.traduce("diccionari.llistat")%></legend>
        <form method="POST" action="/directori/associar-usuaris-grups">
                <input type="hidden" id="form" name="form"
value="com.directori.formularis.AssociarUsuarisGrupsUserForm"/><br/>
                <table border="1" cellpadding="1" cellspacing="0" style="width:100%; border-
collapse:collapse; text-align:center;">
                        <tr style="background-color: #6a6a76; color:white;">

        <td><%=LWFi18nService.traduce("manteniment.associar.usuari")%></td>

        <td><%=LWFi18nService.traduce("manteniment.associar.grup")%></td>
                        </tr>
                        <%
                        ArrayList<AssociarUsuarisGrupsUserForm> llistatAssociar =
(ArrayList<AssociarUsuarisGrupsUserForm>) request.getAttribute("LLISTAT");
                        if (llistatAssociar!=null) {
                                for (int i=0; i<llistatAssociar.size(); i++) {
                                        AssociarUsuarisGrupsUserForm form =
llistatAssociar.get(i);
                        %>
                        <tr style="background-color:white; border-width: 1px; border-spacing:
0px; border-color: black; text-align:left;">
                                <td>
                                <%
                                String usuari = "";
                                ArrayList<UsuarisUserForm> llistatUsuaris =
(ArrayList<UsuarisUserForm>) request.getAttribute("LLISTAT_USUARIS");
                                if (llistatUsuaris!=null) {
                                        for (int j=0; j<llistatUsuaris.size(); j++) {
                                                UsuarisUserForm usu = llistatUsuaris.get(j);
                                                if (form.getIdUsuari().equals(usu.getId())) {
                                                        usuari = usu.getNom();
                                                        break;
                                                }
                                        }
                                }
                                %>
                                <%=usuari%>
                                </td>
                                <td>
                                <%
                                String grup = "";
                                ArrayList<GrupsUserForm> llistatGrups =
(ArrayList<GrupsUserForm>) request.getAttribute("LLISTAT_GRUPS");
                                if (llistatGrups!=null) {
                                        for (int j=0; j<llistatGrups.size(); j++) {
                                                GrupsUserForm gr = llistatGrups.get(j);
                                                if (form.getIdGrup().equals(gr.getId())) {
```

```
                                                grup = gr.getNom();
                                                break;
                                        }
                                }
                        }
                        %>
                        <%=grup%>
                        </td>
                </tr>
                <%
                        }
                }
                %>
        </table>
</form>
</fieldset>
<br/>
```

7.2.3.7 directori/pantalles/grups/cos.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.GrupsUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Formulari -->
<fieldset>
        <legend><%=LWFi18nService.traduce("diccionari.llistat")%></legend>
        <form method="POST" action="/directori/grups">
                <input type="hidden" id="form" name="form"
value="com.directori.formularis.GrupsUserForm"/><br/>
                <table>
                        <tr>
                                <td><%=LWFi18nService.traduce("manteniment.grup.id")%></td>
                                <td><input type="text" id="id" name="id" value=""/></td>
                        </tr>
                        <tr>
                                <td><%=LWFi18nService.traduce("manteniment.grup.nom")%></td>
                                <td><input type="text" id="nom" name="nom" value=""/></td>
                        </tr>
                        <tr>

        <td><%=LWFi18nService.traduce("manteniment.grup.descripcio")%></td>
                                <td><input type="text" id="descripcio" name="descripcio"
value=""/></td>
                        </tr>
                        <tr>
                                <td colspan="2" style="text-align:right;"><input type="submit"
value="<%=LWFi18nService.traduce("manteniment.grup.submit")%>"/></td>
                        </tr>
                </table>
        </form>
</fieldset>
<br/>
```

7.2.3.8 directori/pantalles/grups/llistat.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.GrupsUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Llistat -->
<fieldset>
        <legend><%=LWFi18nService.traduce("diccionari.formulari")%></legend>
        <form method="POST" action="/directori/grups">
                <input type="hidden" id="form" name="form"
value="com.directori.formularis.GrupsUserForm"/><br/>
```

```
		<table border="1" cellpadding="1" cellspacing="0" style="width:100%; border-
collapse:collapse; text-align:center;">
			<tr style="background-color: #6a6a76; color:white;">
				<td><%=LWFi18nService.traduce("manteniment.grup.id")%></td>
				<td><%=LWFi18nService.traduce("manteniment.grup.nom")%></td>

	<td><%=LWFi18nService.traduce("manteniment.grup.descripcio")%></td>
			</tr>
			<%
			ArrayList<GrupsUserForm> llistat = (ArrayList<GrupsUserForm>)
request.getAttribute("LLISTAT");
			if (llistat!=null) {
				for (int i=0; i<llistat.size(); i++) {
					GrupsUserForm usu = llistat.get(i);
			%>
			<tr style="background-color:white; border-width: 1px; border-spacing:
0px; border-color: black; text-align:left;">
				<td><%=usu.getId()%></td>
				<td><%=usu.getNom()%></td>
				<td><%=usu.getDescripcio()%></td>
			</tr>
			<%
				}
			}
			%>
		</table>
	</form>
</fieldset>
<br/>
```

7.2.3.9 directori/pantalles/login/cos.jsp

```
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Cos del login -->
<fieldset>
	<legend><%=LWFi18nService.traduce("login.titulo")%></legend>
	<form method="POST" action="/directori/autenticar">
		<input type="hidden" id="form" name="form"
value="com.directori.formularis.AutenticarUserForm"/><br/>
		<table>
			<tr>
				<td><%=LWFi18nService.traduce("login.usuari")%></td>
				<td><input type="text" id="user" name="user" value=""/></td>
			</tr>
			<tr>
				<td><%=LWFi18nService.traduce("login.password")%></td>
				<td><input type="text" id="password" name="password"
value=""/></td>
			</tr>
			<tr>
				<td colspan="2" style="text-align:right;"><input type="submit"
value="<%=LWFi18nService.traduce("login.submit")%>"/></td>
			</tr>
		</table>
	</form>
</fieldset>
```

7.2.3.10 directori/pantalles/usuaris/cos.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.UsuarisUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Formulari -->
<fieldset>
	<legend><%=LWFi18nService.traduce("diccionari.formulari")%></legend>
	<form method="POST" action="/directori/usuaris">
```

```
		<input type="hidden" id="form" name="form"
value="com.directori.formularis.UsuarisUserForm"/><br/>
		<table>
			<tr>
				<td><%=LWFi18nService.traduce("manteniment.usuari.id")%></td>
				<td><input type="text" id="id" name="id" value=""/></td>
			</tr>
			<tr>
				<td><%=LWFi18nService.traduce("manteniment.usuari.nom")%></td>
				<td><input type="text" id="nom" name="nom" value=""/></td>
			</tr>
			<tr>

	<td><%=LWFi18nService.traduce("manteniment.usuari.cognom1")%></td>
				<td><input type="text" id="cognom1" name="cognom1"
value=""/></td>
			</tr>
			<tr>

	<td><%=LWFi18nService.traduce("manteniment.usuari.cognom2")%></td>
				<td><input type="text" id="cognom2" name="cognom2"
value=""/></td>
			</tr>
			<tr>
				<td><%=LWFi18nService.traduce("manteniment.usuari.mobil")%></td>
				<td><input type="text" id="mobil" name="mobil" value=""/></td>
			</tr>
			<tr>
				<td><%=LWFi18nService.traduce("manteniment.usuari.mail")%></td>
				<td><input type="text" id="mail" name="mail" value=""/></td>
			</tr>
			<tr>
				<td colspan="2" style="text-align:right;"><input type="submit"
value="<%=LWFi18nService.traduce("manteniment.usuari.submit")%>"/></td>
			</tr>
		</table>
	</form>
</fieldset>
<br/>
```

7.2.3.11 directori/pantalles/usuaris/llistat.jsp

```
<%@page import="java.util.ArrayList" %>
<%@page import="java.lang.Exception" %>
<%@page import="com.directori.formularis.UsuarisUserForm" %>
<%@page import="LWF.i18n.LWFi18nService" %>

<!-- Llistat -->
<fieldset>
	<legend><%=LWFi18nService.traduce("diccionari.llistat")%></legend>
	<form method="POST" action="/directori/usuaris">
		<input type="hidden" id="form" name="form"
value="com.directori.formularis.UsuarisUserForm"/><br/>
		<table border="1" cellpadding="1" cellspacing="0" style="width:100%; border-
collapse:collapse; text-align:center;">
			<tr style="background-color: #6a6a76; color:white;">
				<td><%=LWFi18nService.traduce("manteniment.usuari.id")%></td>
				<td><%=LWFi18nService.traduce("manteniment.usuari.nom")%></td>

	<td><%=LWFi18nService.traduce("manteniment.usuari.cognom1")%></td>

	<td><%=LWFi18nService.traduce("manteniment.usuari.cognom2")%></td>
				<td><%=LWFi18nService.traduce("manteniment.usuari.mobil")%></td>
				<td><%=LWFi18nService.traduce("manteniment.usuari.mail")%></td>
			</tr>
			<%
			ArrayList<UsuarisUserForm> llistat = (ArrayList<UsuarisUserForm>)
request.getAttribute("LLISTAT");
			if (llistat!=null) {
				for (int i=0; i<llistat.size(); i++) {
					UsuarisUserForm usu = llistat.get(i);
```

```
                        %>
                        <tr style="background-color:white; border-width: 1px; border-spacing:
0px; border-color: black; text-align:left;">
                                <td><%=usu.getId()%></td>
                                <td><%=usu.getNom()%></td>
                                <td><%=usu.getCognom1()%></td>
                                <td><%=usu.getCognom2()%></td>
                                <td><%=usu.getMobil()%></td>
                                <td><%=usu.getMail()%></td>
                        </tr>
                        <%
                                }
                        }
                        %>
                </table>
        </form>
</fieldset>
<br/>
```

7.2.3.12 directori/plantilles/login.jsp

```
<%@page import="java.lang.String" %>
<%@page import="LWF.util.LogUtil" %>
<%
String missatge = (String) request.getAttribute("TEMPLATE_missatge");
LogUtil.print("TEMPLATE_missatge="+missatge);

String cos = (String) request.getAttribute("TEMPLATE_cos");
LogUtil.print("TEMPLATE_cos="+cos);
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 <html>
        <head>
                <title>Login</title>
        </head>
        <body style="background-color: #6a6a76; margin-top:200px;">
                <center>

                        <div style="text-align:right; width: 300px; color:white;">
                                <span>
                                        <style>
                                                a {color:white; text-decoration:none; font-
weight:bold; font-family: verdana; font-size:12px;}
                                                a:hover {color:#DEDE99;}
                                        </style>
                                        <a href="/directori/login?idioma=CA">CA</a> -
                                        <a href="/directori/login?idioma=ES">ES</a> -
                                        <a href="/directori/login?idioma=EN">EN</a>
                                </span>
                        </div>
                        <div style="text-align:left; padding:20px; background-color: #dede99;
font-family: verdana; font-size:12px; width: 300px; border: 1px solid black;">
                                <jsp:include page="<%=missatge%>"></jsp:include>
                                <jsp:include page="<%=cos%>"></jsp:include>
                        </div>
                </center>
        </body>
</html>
```

7.2.3.13 directori/plantilles/manteniment.jsp

```
<%@page import="java.lang.String" %>
<%@page import="LWF.i18n.LWFi18nService" %>
<%@page import="LWF.util.LogUtil" %>
<%
String cap = (String) request.getAttribute("TEMPLATE_cap");
LogUtil.print("TEMPLATE_cap="+cap);
```

```
String missatge = (String) request.getAttribute("TEMPLATE_missatge");
LogUtil.print("TEMPLATE_missatge="+missatge);

String menu = (String) request.getAttribute("TEMPLATE_menu");
LogUtil.print("TEMPLATE_menu="+menu);

String cos = (String) request.getAttribute("TEMPLATE_cos");
LogUtil.print("TEMPLATE_cos="+cos);

String llistat = (String) request.getAttribute("TEMPLATE_llistat");
LogUtil.print("TEMPLATE_llistat="+llistat);

String peu = (String) request.getAttribute("TEMPLATE_peu");
LogUtil.print("TEMPLATE_peu="+peu);
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 <html>
        <head>
                <title>Directori</title>
        </head>
        <body style="background-color: #6a6a76; margin-top:20px;">
                <center>
                        <div style="text-align:right; width: 600px; color:white;">
                                <span>
                                        <style>
                                                a#logout {color:white; text-decoration:none; font-
weight:bold; font-family: verdana; font-size:12px;}
                                                a:hover#logout {color:#DEDE99;}
                                        </style>
                                        <a id="logout"
href="/directori/login"><%=LWFi18nService.traduce("manteniment.logout")%></a>
                                </span>
                        </div>
                        <div style="text-align:left; padding:20px; background-color: #dede99;
font-family: verdana; font-size:12px; width: 600px; border: 1px solid black;">
                                <jsp:include page="<%=cap%>"></jsp:include>
                                <jsp:include page="<%=missatge%>"></jsp:include>
                                <jsp:include page="<%=menu%>"></jsp:include>
                                <jsp:include page="<%=cos%>"></jsp:include>
                                <jsp:include page="<%=llistat%>"></jsp:include>
                                <jsp:include page="<%=peu%>"></jsp:include>
                        </div>
                </center>
        </body>
</html>
```