Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Introducció a Canigó

Versió 1.0

Barcelona, 21 / febrer / 2006

Històric de modificacions

Data	Autor	Comentaris	Versió
20/01/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0

Llegenda de Marcadors

Índex

1. Introducció

1.1. Context d'Aplicació i Escenaris d'Ús

En tot desenvolupament, sobretot quan es tracta d'una aplicació distribuída, existeix la necessitat de definir una arquitectura basada en la plataforma o llenguatge a utilitzar. Aquesta arquitectura hauria de proporcionar a l'equip de desenvolupament dels serveis necessaris: control d'excepcions, mecanismes de logs, accés a dades, etc. En tots aquests serveis és també necessària la definició d'una capa d'abstracció que ens independitzi en la mesura de lo possible de la implementació escollida.

A més de l'arquitectura, quasi sempre es repeteixen les mateixes necessitats: cóm gestionar les nostres excepcions, cóm enviar un correu electrònic o fins i tot cóm realitzar un llistat de resultats parcial. Tots aquests problemes es poden resoldre de vàries maneres. La forma en la que ho resolguem farà que la solució sigui més o menys costosa, més o menys óptima o i més o menys fàcil de mantenir.

Els desenvolupadors s'enfronten a un llarg ventall d'opcions, des de contenidors lleugers com Spring, NanoContainer o HiveMind fins a frameworks Web com WebWork, JSF, Tapestry o Struts. El nombre de seleccions s'incrementa a mesura que requerim la utilització d'una característica concreta a les nostres aplicacions (connexió FTP, connexió amb una cua, enviament de mails, generació d'un fitxer PDF, etc.). La selecció és complexa i requereix de grans espais de temps (investigació, proves, etc.) per a poder avaluar quina és la millor opció. Canigó facilita aquesta selecció oferint en cadascun dels àmbits d'aplicació una solució adequada, oferint la màxima flexibilització.

Per l'estalvi en temps de desenvolupament així com l'alt grau de funcionalitat i estabilitat del codi proporcionat, és molt recomanable evitar la construcció d'una arquitectura específica per cada projecte concret. Construir un framework de qualitat i estable és complex i molt sovint més costós que desenvolupar una aplicació de negoci. Els frameworks s'han convertit en la base actual de l'enginyeria de sofware, principalment per la gran capacitat de reutilització de codi i la facilitat de generació de noves aplicacions, però la seva selecció, ús i extensió és en el molts casos complexa.

1.2. Per què Canigó

El framework de desenvolupament J2EE Canigó té com a missió integrar i extendre les diferents solucions en cada àmbit d'aplicació (traces, mailing, bases de dades, etc.). L'equip de desenvolupament de Canigó està sempre avaluant noves tecnologies i estàndars, i incorporant-les una vegada han superat els tests corresponents.

Mitjançant Canigó es proporciona un arquitectura:

- De cost reduït
- Flexible i escalable

- Oberta, basada en estàndars i no lligada a cap proveïdor
- Fàcil d'evolucionar, ampliar i adaptar a les necessitats
- Fiable, estable i provada
- D'alt rendiment

A més, permet, entre d'altres:

- Accelar el desenvolupament d'aplicacions J2EE, simplificant el cicle de desenvolupament
- Facilitar l'operació i gestió de les aplicacions generades
- Proporcionar una arquitectura d'aplicacions consistent entre aplicacions, coherents per tots els desenvolupaments
- Minimitzar la corba d'aprenentatge dels programadors

El framework Canigó, a més de l'indicat té com a objectius principals oferir:

- Programació Orientada a Interfícies. Oferir mitjançant interfícies l'accés a la implementació
- Configuració Declarativa. Configurar tots els serveis i elements de l'aplicació de forma declarativa sense afectar al codi
- Solució Oberta. Poder afegir i intercanviar qualsevol peça amb un cost molt reduïy
- Simplificar la complexitat inherent a J2EE
- Oferir components de desenvolupament
- Proporcionar eines de suport que facilitin el cicle de desenvolupament

1.3. A qui va dirigit

Canigó està dirigit a desenvolupadors tant novells com experimentats. En qualsevol cas qualsevol framework requereix del coneixement previ de vàries tecnologies i d'un temps d'adequació que en cap cas es considera superior al del cicle inherent a qualsevol altre framework. En el cas de Canigó cal disposar de nocions bàsiques sobre:

- Tecnologies bàsiques Web: Servlets, JSP, HTML, CSS, JSTL
- Struts
- Spring
- Hibernate

1.4. Glossari

API
Application Programming Interface. Especificació d'una llibreria que documenta la seva interfície i permet la seva utilització sense coneixement dels seus detalls interns.

HTML
Hypertext Markup Language. Llenguatge de tags per la creació de documents per la Web.

Java
Plataforma pel desenvolupament de software que proporciona independència de la plataforma on s'instal·larà l'aplicació, i proporciona una gran quantitat de APIs estàndars.

J2EE
Java 2, Enterprise Edition. Versió avançada de la plataforma Java de Sun Microsystems, destinada al desenvolupament d'aplicacions empresarials.

J2SE
Java 2, Standard Edition. Versió bàsica del conjunt d'eines i APIs de Sun Microsystems per la creació d'aplicacions en la plataforma Java

2. Arquitectura General de Canigó

L'arquitectura de Canigó es basa en l'arquitectura MVC, on existeix un procés d'abstracció que permet dividir una aplicació en components lógics que poden ser creats més fàcilment. L'arquitectura Model-Vista-Control·lador (MVC) és un procés d'abstracció pel qual es divideix la funcionalitat entre els components involucrats en mantenir i presentar les dades, de manera que podem minimitzar la dependència entre ells. Aquesta filosofia permet que diferents equips puguin desenvolupar amb perfils diferenciats, i el més important, permet que per futures direccions (tècniques, aparició de nous estàndars, etc.) el traspàs sigui mínim.

L'arquitectura MVC divideix els objectes implicats en una aplicació en 3 tipus:

1) Model

   Representa les dades de l'aplicació i les regles de negoci (que gobernen en alguns casos l'accés i modificacions a aquestes dades de l'aplicació).

   El Model notifica a les vistes dels seus canvis (totes aquelles vistes que vulguin rebre notificació dels canvis en el seu model associat) i proporciona els mètodes necessaris per a que les vistes puguin consultar el seu model i conéixer l'estat.

2) Vista

   Mostra el contingut d'un model. Accedeix a les dades del model i especifica com s'han de mostrar aquestes dades.

   Quan el model canvia, rep una notificació d'aquest, accedeix a ell i mostra les noves dades

   La Vista també s'encarrega de redireccionar els events/accions de l'usuari cap al Control·lador

3) Control·lador

   Defineix el comportament de l'aplicació, interpreta els events de l'usuari i els mapeja en accions a realitzar sobre el model. Aquestes accions poden ser realitzar un canvi de l'estat del model o bé executar funcions de la lógica de negoci del model.

   Depenent de l'acció realitzada per l'usuari, el Control·lador pot seleccionar una nova vista a mostrar com a resposta a la petició realitzada per l'usuari.

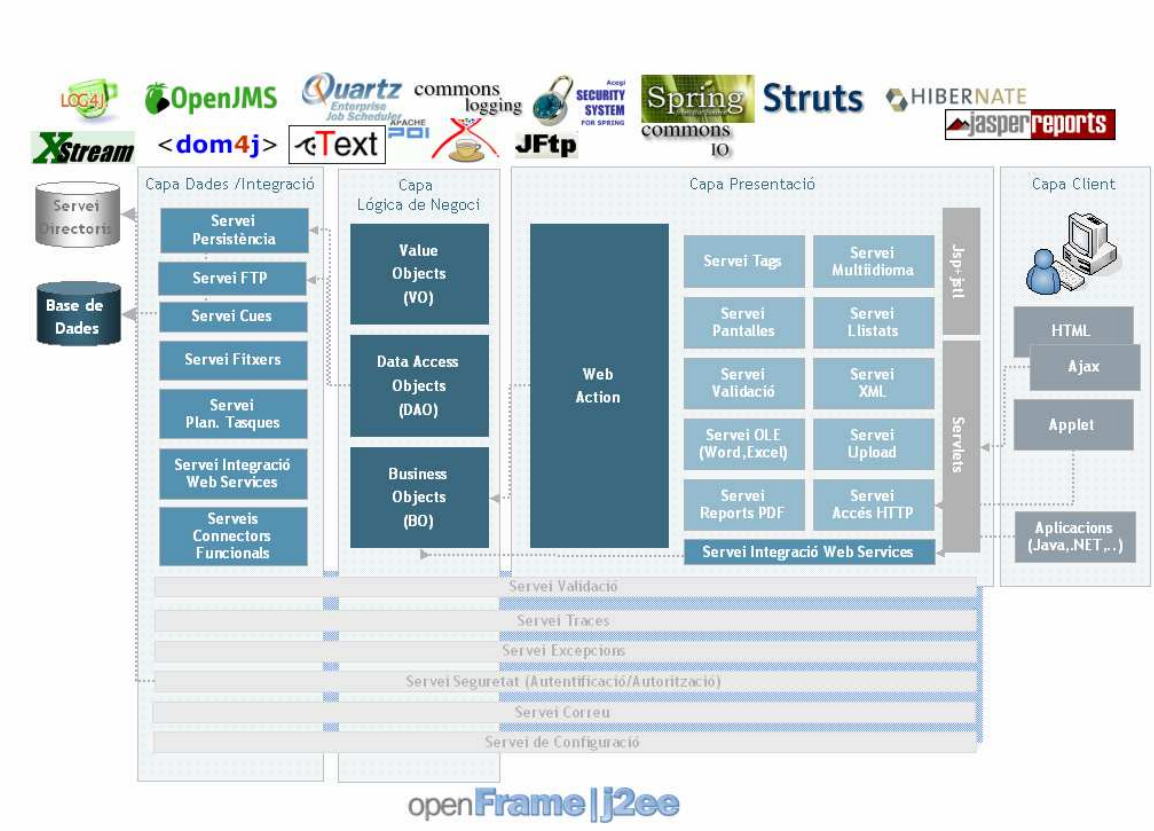
El flux en aquesta arquitectura seria :

- L'usuari introdueix en alguns casos informació a la vista i accepta les dades o prem un botó. La vista envia l'acció realitzada al Controlador

- El Controlador coordina i realitza els canvis al Model com a resposta a l'acció rebuda des de la vista
- El Model canviat avisa a les vistes associades de que ha canviat (aquelles que mostren el Model canviat)
- Les Vistes accedeixen al Model per obtenir les noves dades i s'actualitzen amb aquestes noves dades.
- El Controlador selecciona i mostra la pantalla resultat de la petició

Tammateix a més de l'estructura lógica dels components segons el patró MVC, Canigó defineix la seva ubicació segons una arquitectura en 3 capes:

- Capa de Presentació
- Capa de Lógica de Negoci
- Capa de Dades/Integració



En cadascuna de les capes s'han definit diferents Serveis que es mostren en el gràfic de a dalt. A més dels serveis propis de cadascuna de les capes s'ofereixen Serveis de Propósit General, adequats per qualsevol de les capes. Tots els serveis es troben definits mitjançant interfícies, aillant així de la implementació concreta escollida.

Els diferents serveis es troben paquetitzats en llibreries jars diferents. El gran avantatge és que des dels nostres aplicatius no cal disposar de totes les llibreries, sinó únicament de les necessàries. Ara bé, existeixen alguns serveis de base dels que cal disposar com a mínim per a poder executar qualsevol servei. Aquests són:

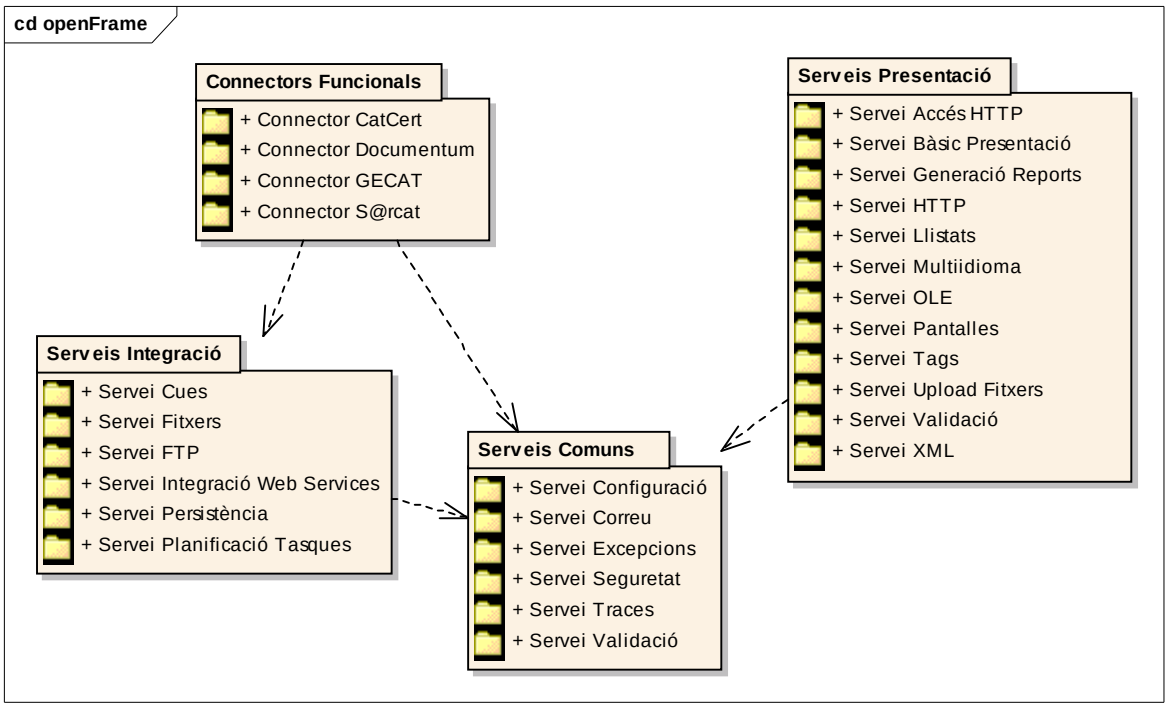
Servei	Llibreria
Servei Core (servei d'utilitats bàsiques)	openFrame-core-XX.jar
Servei d'Excepcions	openFrame-services-exceptions-XX.jar
Servei de Traces o Logging	openFrame-services-logging-XX.jar
Servei de Configuració	openFrame-services-configuration-XX.jar

On XX s'ha de substituir per la versió que es faci servir (en l'actualitat 1.0)

L'arquitectura de Canigó es basa en la integració de Struts amb Spring, aprofitant la potència de la injecció de Spring per definir les nostres accions. Per a conéixer quin és el procés de tractament de les peticions es recomana una lectura prèvia de l'apartat 'Arquitectura i Components - Tractament de les peticions Web' del document 'Serveis de Presentació'.

Si es vol tenir un coneixement en més detall de les possibilitats d'integració entre Struts i Spring es recomana la lectura de les urls' http://www.jroller.com/comments/fdiotalevi/Weblog/two_ways_to_integrate_struts' i 'http://www-128.ibm.com/developerworks/java/library/j-sr2.html'.

En el següent diagrama es mostren els serveis proporcionats per Canigó:



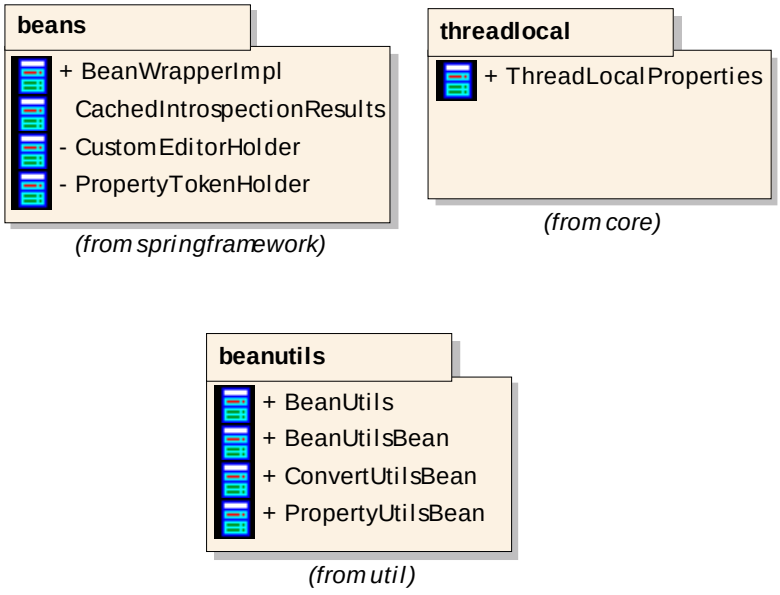
2.1. Serveis de Propósit General

Els Serveis de Propósit General són serveis que poden ser utilitzats des de qualsevol capa de l'aplicació. Dins aquest conjunt es troben:

- Servei de Validació. Permet la validació de les dades (de lógica de negoci)
- Servei de Traces. Permet generar missatges des de l'aplicació que poden ser tractats en la seva posteritat.
- Servei d'Excepcions. Defineix cóm gestionar les excepcions llençades per qualsevol element de l'arquitectura o de l'aplicació
- Servei de Seguretat. Gestió integral de l'autentificació (accés a l'aplicatiu) i de l'autorització (definir quins rols tenen accés a determinada funcionalitat, ...)
- Servei de Correu. Enviament de correu electrónic
- Servei de Configuració. Permet la configuració dels serveis i de les aplicacions de forma externalitzada (mitjançant fitxers de configuració)

2.1.1. Servei Core

El Servei core defineix classes d'utilitat per qualsevol dels serveis Canigó. Es tracten de classes internes utilitzades des dels serveis.



Entre d'altres s'ofereixen les següents funcionalitats (adicionals a les ofertes pels serveis en els que es basen):

- En cas de propietats nested (per exemple: categoria.nom) si s'intenta introduir un valor es llença per defecte un null. A Canigó s'han extés els tractaments proporcionats per

Spring i per Commons BeanUtils perque es crein els objectes intermitjos a mesura que es requereixin.

2.1.2. Servei de Validació

Permet la validació de dades, tant de negoci com de presentació. La definició de les validacions es realitza de forma externalitzada mitjançant fitxers.

Veure el document 'Servei de Validació' per a més referència.

2.1.3. Servei de Traces

Les traces permeten detectar i localitzar amb més facilitat errors de l'aplicació, entrada de dades incorrectes segons la lògica del sistema, i inclús realitzar un seguiment de qui ha fet una determinada operació per a portar a terme una **auditoria.**

El Servei de Traces permet, entre d'altres:

- Definir nivells de traces (d'informació, fatals, errors, etc.)
- Definir en quines sortides es generarà la traça: consola, fitxers, base de dades, correu electrónic, etc.
- Canviar en qualsevol moment quin és el mínim nivell de traces que volem mostrar, sense haver d'afectar a les classes que generen les traces
- Definir el format de sortida de les nostres traces: incorporar l'hora, el número de línia del codi on s'ha produit i la seva classe, etc.
- Incorporar informació de context a la traça: usuari, ip del client, etc.

Veure document 'Servei de Traces' per a més informació.

2.1.4. Servei d'Excepcions

La gestió d'excepcions permet informar de que s'ha produït un error al realitzar una petició. Aquest error podrà ser tractat adecuadament i en cas necessari informar a l'usuari, llençar una traça, enviar un correu, etc. L'ús apropiat de les excepcions fa que els nostres aplicatius siguin més robusts, més fàcils de desenvolupar i mantenir, més lliures d'errors i més fàcils de utilitzar. Per aquest motiu és important que donem els màxims detalls en les excepcions.

El Servei d'Excepcions permet, entre d'altres:

- Afegir els detalls necessaris a les nostres excepcions mitjançant l'ús de la classe 'ExceptionDetails'

  Aquesta classe ens permetrà definir:

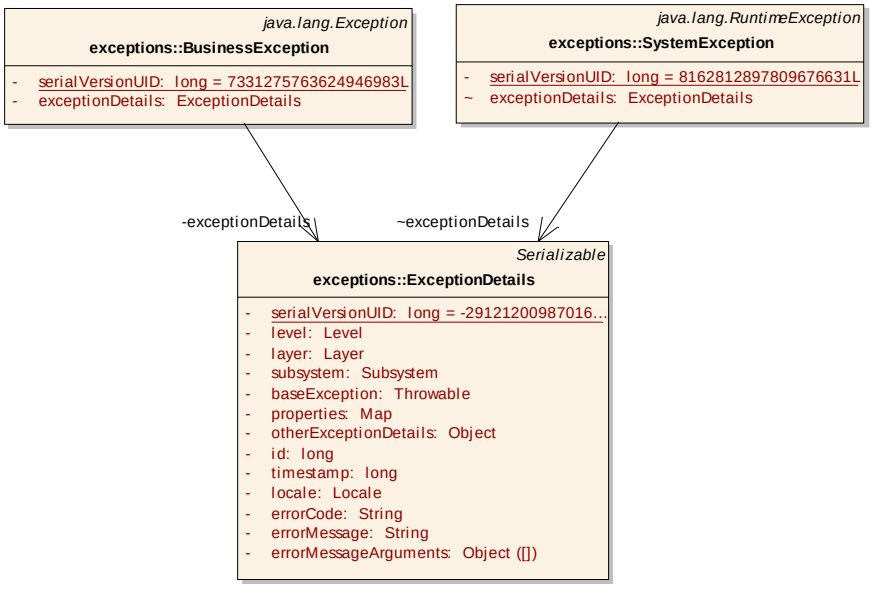
  1) Subsistema en el que s'ha produït l'excepció
  2) Capa en la que s'ha produït l'excepció. Ens permetrà conéixer a quina alçada s'ha produit.

3) Nivell de l'excepció (FATAL, ERROR, WARNING, INFO,...). Aquest nivell ens permetrà definir la criticitat de l'excepció i per tant el seu tractament diferenciat
4) Codi de l'error. Mitjançant aquest atribut obtenim que les nostres excepcions siguin **internacionalitzades** de forma que el missatge que es mostri a l'usuari depengui del seu idioma escollit
5) Arguments del missatge d'error (a incorporar en el moment de la traducció com a paràmetres del missatge)
6) Timestamp. Moment en el que s'ha produit l'excepció
7) Properties. Mapa per introduir paràmetres adicionals a l'excepció
8) BaseException. Aquest atribut ens permet anidar l'excepció causa que ha provocat l'excepció llençada

- L'ús de 2 excepcions diferenciades: BusinessException i SystemException

La primera excepció hereta de la classe 'Exception' i serà usada des de les aplicacions per llençar excepcions de negoci. Permet afegir els detalls de l'excepció.

El segon tipus d'excepció (hereta de RuntimeException) serà utilitzat pels serveis de Canigó. L'avantatge és que no cal que siguin capturades per les classes de l'aplicació (evitant així un dels desavantatges inherents al llenguatge Java i el tractament obligatori d'excepcions).



- El tractament d'excepcions de forma horitzontal mitjançant la definició de gestors.

Mitjançant l'ús de Aspect Oriented Programming podem definir cóm es tractaran les nostres excepcions de forma global. Així, podem configurar que en el moment de que

una classe llenci una excepció d'un determinat tipus es faci una traça (mitjançant el Servei de Traces per exemple) indicant què ha succeit.

Veure document 'Servei d'Excepcions' per a més informació.

2.1.5. Servei de Configuració

El Servei de Configuració té com a objectiu la configuració de qualsevol element de l'aplicatiu i dels serveis de forma externalitzada.

El seu ús es basa en la utilització dels contexts i factories de Spring. Degut a que la tecnologia utilitzada per aquest contenidor no és intrusiva, es pot canviar de servei de configuració sense afectar a les classes ja creades.

2.1.6. Servei de Correu Electrónic

Aquest servei permet l'enviament de correus electrònics a una o diverses adreces. Es poden utilitzar continguts en text pla, en HTML i incorporar fitxers anexes al correu.

Veure document 'Servei de Correu Electrónic' per a més informació.

2.2. Serveis de Presentació

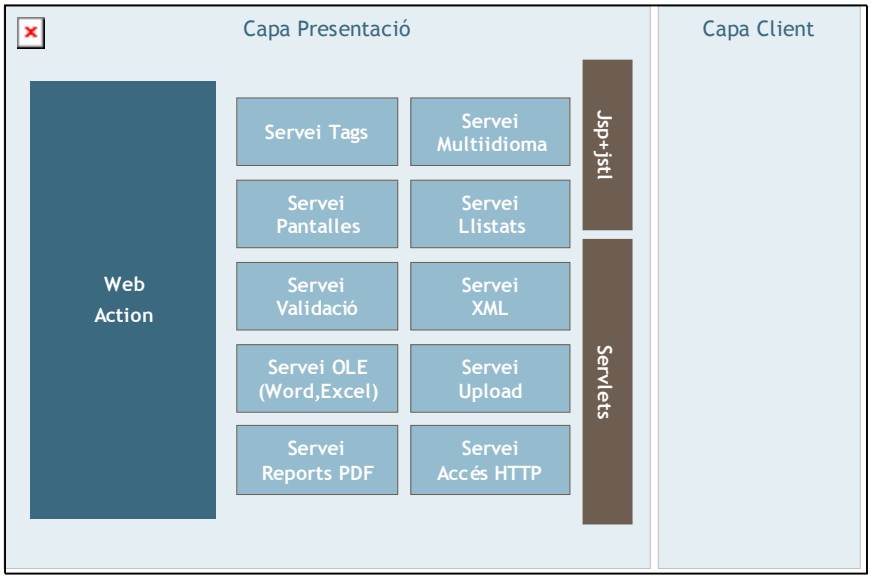
El Servei de presentació té com a objectius:

1) Proporcionar la infrastructura necessària per a desenvolupar aplicacions Web
2) Proporcionar components de presentació (basats en JSP) per a mostrar informació a l'usuari, validar-la i retornar noves dades a l'usuari
3) Proporcionar els mecanismes d'integració i comunicació des de diferents tipus de clients
    - HTML (s'executa en el navegador Web)
    - Ajax. Permet la generació d'aplicacions HTML RIA (Rich Internet Applications)
    - Applets (s'executat dins el navegador Web o de forma externa)
    - Aplicacions

En el punt central de l'arquitectura de la capa de presentació es troben els 'Action'. Aquests són els gestors de les peticions rebudes i que són definits per cada aplicació. Aquesta acció pot requerir de funcionalitat adicional per presentar la informació a l'usuari:

- Servei Tags. Si es fan servir pàgines de presentació a l'usuari cal disposar de components o tags a les pàgines
- Servei Multiidioma. Si l'aplicació ha de ser dirigida a usuaris amb diferents idiomes l'aplicació ha de poder ser internacionalitzada
- Servei de Pantalles, si es requereix de la definició de pantalles amb un look&feel comú

- Servei de Llistats, si es volen mostrar llistats parcials a l'usuari amb capacitat d'ordenar-los per columna, accedir a una pàgina concreta dels resultats o generar formats més adients per al seu tractament (PDF, Excel)
- Servei de Validació, si cal realitzar validacions a les entrades de l'usuari (dates, rangs, etc.) sense que el desenvolupador requereixi del coneixement de llenguatge usat per la validació (Javascript, ...)
- Servei XML, si cal parsejar un fitxer XML rebut i/o ser transformat en objectes del negoci per tal de comunicar-nos amb la capa de negoci.
- Servei OLE, si volem generar o llegir fitxers Word o Excel
- Servei de Upload, per si volem obtenir un fitxer pujat per l'usuari de forma senzilla i tractar-lo
- Servei de Reports PDF, si volem generar un document PDF formatejat i que incorpori les dades rebudes des de la capa de negoci
- Servei Accés HTTP, si volem donar accés a la capa de negoci des de clients atípics com Applets sense afegir cap llibreria adicional al client
- Servei de Integració WebServices, si volem donar accés a la capa de negoci des dels clients mitjançant WebServices



A continuació s'ofereix un resum dels objectius i característiques de cada servei:

2.2.1. Servei de Tags

Proporciona tags per emprar a les planes JSP. Entre d'altres es proporcionen les següents possibilitats:

- Formularis i components amb modus (creació, edició o consulta amb la mateixa plana)
- Indicació de camps requerits automàtica a la plana (si s'ha configurat que el camp és requerit en el servei de validació)

- Camps de tipus data amb assistent calendari
- Conversions automàtiques de dades (eliminació de blancs, pas a majúscules)
- Tooltips d'ajuda associada
- Camps d'edició rica HTML (amb negreta, justificats,...)
- Llistats parcials de dades amb navegabilitat, generació de Excels i PDFS
- Accés directe als components per teclat
- Tabulació automàtica en superar el màxim de caràcters d'un component
- Obtenció i presentació en background (sense refrescar la pàgina) dels valors d'un component de selecció segons el valor escollit a un altre component

2.2.2. Servei Multiidioma

El Servei de multiidioma té com a objectiu el desenvolupament d'aplicacions multiidioma en el que no sigui necessària cap reenginyeria cada vegada que s'incorpori un nou llenguatge a l'aplicació.

Mitjançant la definició de fitxers internacionalitzats es permet la integració des dels tags i des de les classes de presentació.

2.2.3. Servei de Pantalles

Tota aplicació Web requereix d'un Look&Feel (L&F) comú. Si el Look&Feel es defineix a cada pàgina de forma manual, un canvi de L&F a l'aplicació provocaria la necessitat de modificar una a una cadascuna de les pàgines de l'aplicació. En un bon disseny és important que siguem capaços de separar el L&F del contingut de les pàgines.

El Servei de Pantalles permet definir l'estil comú, especificant per exemple que usarem una capçalera, un cos, un menú a l'esquerra i un peu. Les pàgines de l'aplicació no haurien de definir aquests parts, només haurien de preocupar-se de definir el seu contingut. La pàgina final es construirà de forma dinàmica fent ús de les parts comunes i de les parts específiques de la pàgina.

El Servei de Pantalles permet:

- La creació de pantalles mitjançant l'ensamblatge de vàries parts
- Definir herència entre pantalles, pel que podem definir el cas general i les seves particularitats
- Definir diferents tipus de pantalles: vertical, portal, horitzontal

2.2.4. Servei de Llistats

El servei de llistats permet la presentació de llistats parcials HTML en el que es permet a l'usuari:

- Ordenar per columna de manera ascendent o descendent.
- Paginació dels resultats i navegació per pàgines mitjançant "avanç" i "retrocés" (primera, última, següent, anterior)
- Presentar un número determinat de resultats per pàgina.

- Generació de PDF i Excel del llistat

A més, proporciona ajut a l'usuari, presentant:

- Rollover de la fila al passar el ratolí per sobre.
- Desactivació dels botons de navegació depenent de la pàgina en la que s'estigui ubicat.
- Indicació de la columna per la que s'està ordenant mitjançant icones, mostrant si és ascendent o descendent.
- Look & feel configurable.

2.2.5. Servei de Validació

En la gestió de les peticions és necessària, per lo general, una validació prèvia de les dades entrades per l'usuari. Aquesta validació inclou, entre d'altres:

- Comprovar que s'han introduit dades en un camp definit com a obligatori
- Comprovar que la dada introduïda es pot transformar en un tipus definit (enter, flotant, data,...)
- Comprovar que la dada compleix una expressió regular (correu electrónic, nif, etc.)

La validació realitzada a la presentació té com a objectiu filtrar les dades entrades abans de que arribi a la lógica de negoci.

El Servei de Validació de Presentació es basa en l'extensió del Servei de Validació de la capa de serveis general. La diferència entre un i l'altre radica principalment en que el de presentació tracta amb paràmetres de tipus text, mentre que el segon pot validar dades de qualsevol tipus.

Per exemple, quan l'usuari introdueix una data aquesta arriba a la capa de presentació mitjançant un paràmetre HTTP. Aquest és de tipus String. En canvi, en el negoci, per lo general, aquesta data serà de tipus 'java.util.Date'. Com és fàcil de constatar un valor de tipus 'java.util.Date' ja és una data correcta (no pot haver-hi dins una cadena 'AAA', per exemple). En aquest cas, la lógica de negoci no haurà de validar res en quant aquest valor, en canvi, la presentació, per passar a la lógica de negoci una data (java.util.Date) haurà de transformar la dada rebuda. Aquesta dada rebuda, per tant haurà de ser comprovada i en aquest cas farà servir el Servei de Validació de Presentació.

El Servei de Validació permet definir quines validacions cal realitzar en els camps mitjançant l'ús de fitxers externs XML. El Servei de Tags, en comunicació amb aquest generar el codi Javascript necessari per validar les dades al navegador de l'usuari.

2.2.6. Servei XML

El Servei XML té com a objectiu proporcionar les bases de la manipulació de fitxers XML.Està orientat a 2 possibilitats diferenciades:

- Manipulació de documents XML.

Poder crear i modificar fitxers XML. Permet traversar de forma simple i directa a determinats parts del fitxer.

- Serialització d'objectes en fitxers XML
  Permet generar a partir d'objectes estructures XML. Tammateix permet recuperar a a partir d'un fitxer XML un conjunt d'instàncies d'objectes

Veure document 'Servei XML' per a més informació.

2.2.7. Servei OLE

El servei de manipulació de formats OLE (Excel i Word) permet llegir, crear, modificar i mostrar documents de Microsoft amb formats basats en OLE (Object Linking and Embedding).

Per a més referència consultar el document 'Servei OLE'.

2.2.8. Servei de Upload de Fitxers

Aquest servei permet poder tractar els fitxers que hagin estat adjuntats dins un formulari HTML per part de l'usuari. Una vegada rebut el fitxer aquesta pot ser desat a la base de dades, emmagatzemat al sistema de fitxers o fins i tot tractat mitjançant el Servei XML o el de fitxers.

Per a més referència consultar el document 'Servei de Càrrega de Fitxers'.

2.2.9. Servei de Reports (PDF i altres)

El servei de reporting de Canigó permet integrar d'una manera senzilla dades des de fonts heterogènies i mostrar-les en diferents formats: PDF, XLS, CSV i HTML, dins d'un marc homogeni de desenvolupament.

Per a més referència consultar el document 'Servei de Reports'.

2.2.10. Servei d'accés HTTP

El propósit d'aquest servei és proporcionar les bases de connectivitat entre applets i les capes de negoci. La diferència amb el Servei d'Integració de Web Services és que proporciona un protocol molt lleuger necessari en applets i aplicacions de móbils en el que la instal·lació de llibreries adicionals no és recomanable.

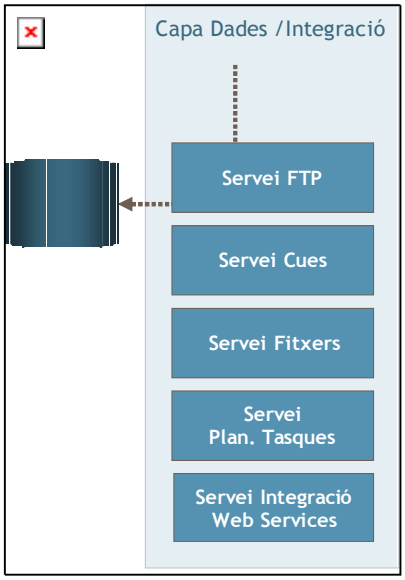
Per a més referència consultar el document 'Servei d'accés HTTP'.

2.3. Serveis d'Integració

Els Serveis d'Integració tenen com a objectiu principal l'accés a aspectes infrastructurals (tant sistemes externs com interns), en el que és necessari definir el protocol d'integració. Dins els serveis d'integració es defineix:

- Servei de Persistència. Ús de bases de dades relacionals sense haver de realitzar tractaments específics de pas d'objectes de negoci a taules.

- Servei FTP. Permet l'enviament de fitxers mitjançant el protocol FTP
- Servei de Cues. Permet la integració amb sistemes de cues basats en JMS
- Servei de Fitxers. Permet accedir al sistema de fitxers per crear directoris, esborrar fitxers, etc.
- Servei Planificació de Tasques. Permet diferir l'execució de determinats mètodes o classes del negoci
- Servei Integració Web Services. Permet la integració amb sistemes externs definits mitjançant Web Services (protocol SOAP).



2.3.1. Servei de Persistència

El Servei de Persistència permet connectar de forma transparent pel programador els objectes de negoci amb les taules de base de dades que els representen. El desenvolupador no ha de programar cap línia de codi SQL.

S'ofereixen, entre d'altres les següents característiques:

- Poder definir cóm es realitzarà el mapeig automàtica entre les taules i associacions de la base de dades amb els objectes de negoci
- Accés mitjançant crides natives en casos específics
- Caché dels objectes prèviament accedits, optimitzacions,etc.
- Suport a vàries bases de dades

Per a més referència consultar el document 'Servei de Persistència'.

2.3.2. Servei FTP

El servei de FTP (File Transfer Protocol) de Canigó permet enviar i rebre arxius, o directoris sencers, des del servidor on s'executa l'aplicació a altres servidors.

Permet configurar les següents característiques:

- Definir el nombre màxim de connexions FTP obertes a la vegada
- Quin temps ha d'haver-hi entre 2 transferències
- Nombre màxim de reintents
- etc.

Per a més referència consultar el document 'Servei FTP'.

2.3.3. Servei de Cues

El Servei de Cues permet configurar i usar de forma senzilla la infraestructura de missatgeria estàndard J2EE (JMS: Java Messaging Service).

Per a més referència consultar el document 'Servei de Cues'.

2.3.4. Servei de Fitxers

El Servei de Fitxers permet la gestió d'un sistema de fitxers, proporcionant entre d'altres les següents característiques:

- Creació, esborrat i cópia de fitxers
- Obtenció d'informació de fitxers (path, nom, extensió, etc.)
- Obtenció de la informació d'un fitxer mitjançant fluxes d'entrada i sortida

Per a més referència consultar el document 'Servei de Fitxers'.

2.3.5. Servei de Planificació de Tasques

L'objectiu del servei es poder configurar un programador de tasques que permeti arrencar determinats processos Java en els moments en que es determini. Es pot determinar que una tasca:

- s'executi en un moment determinat del dia (en precisió de milisegons)
- s'executi en determinats dies de la setmana, del mes o de l'any
- no s'executi en determinats dies
- es realitzi un nombre concret de repeticions
- fins una data concreta o indefinidament
- etc.

El servei garantitza l'execució de la tasca encara que hi hagi una caiguda del sistema, doncs preserva l'estat de les tasques mitjançant base de dades, de forma que controla en tot moment quin és l'estat de les tasques i la seva finalització.

Per a més referència consultar el document 'Servei de Planificació de Tasques'.

2.3.6. Servei d'integració amb WebServices

Aquest servei permet configurar i usar de forma senzilla la infraestructura de Web Services en dues modalitats:

- Exportació de serveis Java mitjançant Web Services.
- Importació de Web Services externs. Es permet la generació de classes Java d' invocació.

L'enfoc d'aquest servei és simplificar la definició de Web Services a partir de serveis Java simples (que no tindran dependències amb la implementació particular de Web Services) i facilitar la invocació a Web Services externs.

Per a més referència consultar el document 'Servei Integració Web Services'.

2.4. Connectors Funcionals

2.4.1. Connector CatCert

El connector CATCERT ens permet connectar amb el sistema CatCert de validació de documents i certificats.

Es proporcionen dos serveis relacionats amb la certificació digital de documents. D'una banda, tenim un servei de validació de certificats digitals i per l'altrea un servei de validació de documents signats dins d'un PKCS7 o validació d'un document amb la signatura a part.

2.4.2. Connector Documentum

Aquest servei consistieix en la capacitat d'emmagatzemar i recuperar documents del repositori documental.

Tindrem les següents funcionalitats per suportar aquestes operacions :

- Establiment de sessió / Desconnexió de sessió.
- Incorporació d'un nou document al repositori.
- Recuperació de documents del repositori a partir de criteris de cerca.
- Obtenció d'un document a partir del seu identificador.

2.4.3. Connector GECAT

Aquest connector enllaça amb l'aplicació GECAT, que és el sistema de gestió econòmica corporatiu de la Generalitat basat en SAP. Permet a les aplicacions que es construeixin amb el framework realitzar crides a tota una sèrie de funcionalitats SAP.

Les funcions bàsiques ofereix aquest connector són:

- Execució de funcionalitats on-line exposades per GECAT
- Tramesa a GECAT de documents comptables en mode batch

El connector permet l'accés al SAP mitjançant objectes de consulta. El connector transforma aquests objectes en una cadena de caràcters vàlida per al SAP. La cadena de retorn és transformada novament en un objecte que conté els registres de retorn.

S'exposen les següents funcionalitats a nivell on-line:

- Alta de factures on-line
- Alta d'abonaments generals (factures negatives)
- Alta de factures d'habilitats
- Alta de reserves
- Consulta de partida pressupostària
- Consulta de creditor
- Consulta de document
- Consulta de factura
- Consulta de territori

S'exposen les següents funcionalitats a nivell batch:

- Alta de factures negatives
- Alta documents pressupostaris de despesa (Documents R, A, D, O, O amb proveïdor CPD, MPE)
- Extracció dades de partides pressupostàries
- Extracció general de dades de documents.

2.4.4. Connector S@rcat

El connector S@rcat ens permet connectar amb el Sistema de registre corporatiu d'entrada i sortida S@rcat.

Les funcions bàsiques ofereix aquest connector són:

- Execucio de funcionalitats on-line encapsulant crides a serveis webservices existents a S@RCAT.
- Utilització del connector de S@RCAT per l'enviament de documents a registrar mitjançant FTP.

La disponibilitat del servei on-line del connector dependrà de la disponibilitat horària del webservices del S@rcat.

S'exposen les següents funcionalitats a nivell on-line:

- Login
- Cerca d'assentaments
- Consulta d'assentaments
- Recollir assentaments safata entrada
- Es presortida
- Insertar assentament d'entrada
- Insertar assentament de sortida
- Insertar assentament de safata
- Insertar assentament de presortida
- Obtenir rang de numeració lliure de registres
- Canvi número d'expedient
- Logout

S'exposen les següents funcionalitats a nivell batch:

- Cerca d'assentaments històric
- Llistar taula mestra
- Funcionalitats on-line d'assentaments

3. Base Tecnològica

La base tecnológica de Canigó inclou un gran ventall de tecnologies. Degut a aquesta amplitud es recomana la lectura de cadascun dels serveis per a més informació.

En qualsevol cas, es pot considerar com a base principal de l'arquitectura:

- Ajax pel desenvolupament d'aplicacions RIA (Rich Internet Application)
- Struts, com a framework de presentació
- Spring Framework com a aplicació del control d'inversió
- Hibernate com a mapejador d'objectes a relacional
- AOP (Aspect Oriented Programming) per intercepció de codi

A continuació s'ofereix una breu introducció a cadascuna d'aquestes tecnologies.

3.1. Spring

Spring és un framework que aporta les següents característiques:

- És un contenidor centralitzat d'objectes i serveis, totalment configurable mitjançant fitxers XML
- A través de l'ús de la inversió de control, en particular, la injecció de dependències, permet la configuració d'objectes fora del codi de l'aplicació (el contenidor s'encarrega de la instanciació) i de manera no intrussiva (els objectes configurats no estan lligats a Spring ni han de conéixer les seves classes)

  Per a més detall es poden consultar les següents referències:

http://www.theserverside.com/articles/article.tss?l=IOCBeginners
http://www.martinfowler.com/articles/injection.html

- Redueix el codi d'aplicació dedicat a configurar i localitzar recursos (JNDI, JTA, ...) a l'encarregar-se Spring. El codi de l'aplicació es fa doncs més legible al tenir principalment lógica d'aplicació
- Facilita best practices com programar contra interfícies enlloc de contra classes.

  Això promou el desacoblament de serveis (cal pensar en els objectes de l'aplicació com serveis, que expressen la seva funcionalitat com interfícies i abstraeixen els seus detalls de la configuració de la vista del programador) i facilita el canvi d'una implementació concreta a una altra

  A més la programació amb interfícies facilita les proves unitàries (doncs no és necessari disposar d'un contenidor de EJBs i els serveis es poden simular fàcilment amb MockObjects).

- Gestió de transaccions sense la utilització de APIs específiques mitjançant l'ús de Aspect Oriented Programming (AOP també es pot usar en altres serveis com la gestió de traces, seguretat o excepcions)

3.2. Struts

En l'escenari web el medi d'intercanvi d'informació és el protocol HTTP. Aquest protocol, dissenyat originalment per a una transferència de fitxers hipertext no proporciona cap facilitat per la creació d'aplicatius web.

Struts és un *framework* per a la creació d'aplicatius web basat en el patró de disseny MVC (Model-Vista-Controlador). Proveeix una série de facilitats que permeten concentrar-se en la part de la lògica i el control de l'aplicatiu i deixar de banda altres aspectes mes mecànics, que porta implementats. Proporciona les següents facilitats:

- Un Controlador principal ja implementat (segueix el patró Front Controller[1] i Dispatcher View[2]). Aquest controlador redirigeix la petició a un classe Action concreta (la configurada per tractar la url rebuda)
- Gestió automàtica dels formularis amb manteniment de les dades enviades entre peticions
- Facilitats de presentació de missatges (errors, validacions, etc.)
- Internacionalització de l'aplicatiu
- Llibreria de tags

El funcionament d'un aplicatiu Struts està molt lligat a la separació que proposa el patró MVC unit a la rigidesa de les aplicacions web. Per una part tenim els models, que implementen en el seu codi la lògica i les operacions que hi ha a l'aplicatiu. Per altra banda tenim els controladors, que gestionen tot el fluxe de l'aplicatiu i donen les ordres de què cal fer als models. Els controladors s'executen com una acció implícita de l'usuari sobre alguna Vista i són els iniciadors de l'execució de les funcionalitats que ofereix l'aplicatiu. La Vista es el medi mitjançant el qual l'usuari pot donar ordres al sistema i veure presentats els resultats.

Per tractar els controladors i els formularis de les vistes Struts defineix respectívament els **Action** i els **ActionForm**). Els **Action** són els controladors de l'aplicatiu.Tenen una estructura rígida (hereten d'una classe definida al respecte) en el que bàsicament s'encarreguen de processar la entrada, normalment un formulari i executar les funcionalitats implementades en altres classes. Al final, depenent del resultat de l'execució, redireccionen el fluxe cap una Vista o cap un altra. L'usuari reb aquesta Vista com una pàgina en el seu navegador.

Els **ActionForm** són classes encarregades d'emmagatzemar temporalment el contingut dels camps d'un formulari provinent d'una Vista. Struts automàticament crea una classe del formulari adient, omple tots els atributs amb el valors provinents del formulari i deixa la classe

1 http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html

2 http://java.sun.com/blueprints/corej2eepatterns/Patterns/DispatcherView.html

accesible. A partir d'aquí els Action poden tractar el formulari d'una manera "còmoda" (com un objecte) i en el cas de que calgui tornar a mostrar la Vista original (per exemple si el formulari no ha estat omplert correctament) Struts automàticament mostrarà els camps inicialitzats amb els valors introduïts anteriorment. Aquesta part requereix que a la Vista s'hagin fet servir els tags especials de Struts, que són els que en últim terme implementen aquesta funcionalitat.

Per a que aquest procés funcioni correctament cada part ha de estar implementada seguint el que estableix Struts i tot ha de estar lligat al fitxer de configuracio de Struts: Struts-config.xml.

Com a desavantatges principals podem establir que:

- No estableix la política de translació des dels paràmetres de la petició (HttpServletRequest) a objectes Java comuns

  Tots els paràmetres d'una petició HTTP són text. Les aplicacions Web basades en Struts han d'encarregar-se de fer la translació o parseig d'aquestes dades en objectes del domini.

- Els ActionForm són estructures innecessàries per expressar el model

  Les classes 'ActionForm' no són pensades per representar el model actual. Són pensades com un buffer de validació preliminar entre el format pla de la petició HTTP i les classes del model. Per exemple, un ActionForm pot validar que un paràmetre d'entrada pot ser transformat en Integer, mentre que el model validaria que aquest Integer es troba dins un rang específic. Els ActionForm només poden incloure elements de tipus 'String' i 'boolean'.

  Degut a que els ActionForm i el model existeix en paral·lel es crear un paral·lelisme massiu innecessari. Aquest punt es troba en connexió amb l'anterior punt. Si existissin polítiques de translació dels paràmetres HTTP a objectes de varis tipus, es podria fer servir de forma directa els objectes del model.

- Els Actions han de ser thread-safe

  Els Actions són sempre reubicats a la cache i reutilitzats per Struts. Es tracta d'una optimització de rendiment, però a la vegada una restricció, ja que no es pot emmagatzemar informació en atributs de la classe. Aquesta ha de ser passada entre paràmetres de les funcions.

Canigó solventa aquests desavantatges de forma transparent dins les seves classes bàsiques Web.

3.3. Hibernate

Hibernate proporciona les següents característiques:

- Un mapeig objecte-relacional flexible (taula per classe, múltiples objectes per registre, múltiples taules per objecte, relacions de tipus 1-N, N-M, etc.)
- Persistència dels objectes de forma transaparent. No imposa l'ús d'interfícies o classes concretes, només l'ús de classes estàndar Java (collections, arrays,...)
- Un llenguatge de queries independent de la base de dades (denominat HQL)
- La possibilitat d'accedir de forma nativa tradicional (T-SQL, PL-SQL, ...) i crides a la lógica de la Base de Dades (stored procedures, packages, ...)
- Tota la configuració (mapeigs, queries HQL, queries natives,...) pot definir-se en fitxers de configuració XML
- Cache (multi-layer, threadsafe, non-blocking, clusterable)
- Altres optimitzacions (lazy initializacion, subselect fetching,...)
- Integració amb les tecnologies J2EE (EJB 3.0, JMX, JTA,...)
- Extensibilitat (nous dialectes SQL per altres bases de dades, generadors propis de claus,...)
- Suport a múltiples bases de dades (Oracle, DB2, Sybase, MS SQL Server, PostgreSQL, MySQL, HypersonicSQL, SAP DB, Interbase, Ingres, Informix)

3.4. Ajax

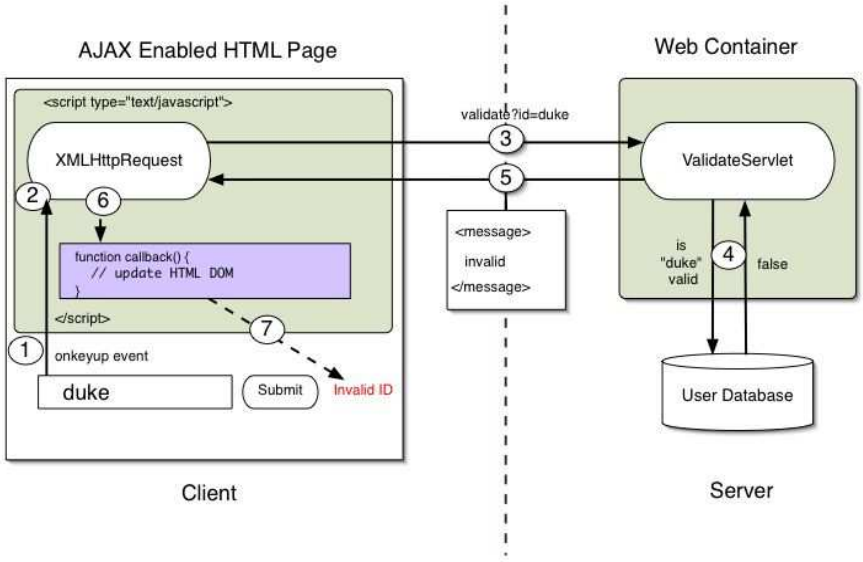
Ajax, o Asynchronous JavaScript i XML, descriu una tècnica de desenvolupament Web per la creació d'aplicacions Web interactives usant una combinació de:

- HTML i CSS
- DOM (Document Object Model) per interactuar de forma dinàmica amb el model HTML presentat
- XMLHttpRequest per interactuar i manipular les dades de forma asíncrona amb el Servidor Web
- Javascript com a part integradora de totes aquestes parts

En el model tradicional Web, tota acció de l'usuari llença una petició HTTP al servidor. El servidor realitza les funcions demanades (obtenció de valors de la base de dades, integració amb serveis, etc.) i finalment retorna una pàgina HTML al navegador client. Mitjançant aquest model no es poden realitzar aplicacions interactives, ja que quan el servidor realitza la petició l'usuari està esperant resposta (no pot interactuar amb la pantalla de cap manera, aquesta es troba bloquejada) i al finalitzar la resposta la pàgina és refrescada (perdem doncs la ubicació de la pantalla en la que ens trobàvem).

Mitjançant Ajax podem realitzar aplicacions Web com si utilitzessim aplicacions natives a la màquina de l'usuari, doncs quan s'interactua amb el servidor només es transfereixen petites quantitats de dades en els 2 sentits (no s'envia la pàgina HTML com en el model tradicional). Com a resultat s'actualitzen només determinades parts de les pàgines, enlloc de refrescar tota la pàgina.

És important indicar que Ajax no és cap tecnologia sinó una técnica. En realitat no es tracta de cap tecnologia nova, doncs es basa en la utilització de l'objecte XMLHttpRequest. Ara bé, s'han establert patrons de disseny que permeten un ús més adequat.



L'ús directe d'aquest objecte és una tasca poc amigable, pel que existeixen llibreries que faciliten l'ús d'aquest objecte.

Canigó fa ús en el seu Servei de Presentació i de Tags de les següents llibreries:

- Prototype
- Behaviour
- DWR
- Ajax Tags

3.4.1. Prototype

La llibreria Prototype és una llibreria bàsica Ajax que permet la realització de vàries tasques bàsiques necessàries en Ajax. Ofereix entre d'altres:

- Classes de tractament de peticions al servidor
  - La classe 'AJAX.Request' per realitzar una petició Ajax
  - La classe 'AJAX.Update' per actualitzar una part del document
  - La classe 'AJAX.PeriodicUpdate' per actualitzar una part del document en cada repetició de l'interval especificat.
- La funció $() com a forma suscinta de la funció comuna 'document.getElementById()'
- Extensió del model Javascript per a poder utilitzar Javascript com un model orientat a objectes (amb definició de classes, mètodes i herència)
- Definir un model de vista-controlador a les nostres pàgines mitjançant la incorporació de listeners als components HTML. Així per exemple, si en el model clàssic usaven:

<input type='checkbox' id='idCheckbox' onclick="...">

Amb aquesta llibreria podrem incorporar el tractament de l'event de forma separada

```
initialize: function(chkBox) {
        this.chkBox = $(chkBox);
        //assignació del métode tractament a l'event rebut
        this.chkBox.onclick = this.showMessage.bindAsEventListener(this);
},
```

En el codi a dalt mostrat, la funció de tractament de l'event és 'showMessage'

- La classe 'AJAX.Insertion' per inserir i recuperar contingut en un document HTML
- La classe 'Form' i 'Field' per realitzar funcions sobre formularis i camps del document

Per més referència consultar la documentació de la llibreria a 'http://www.sergiopereira.com/articles/prototype.js.html#Ajax.Request'

Punts de utilització a Canigó:

- Servei de Tags. Incorporació dinàmica del codi de Calendari
- Servei de Validació. Amb l'ús conjunt de DWR (veure a continuació) per utilitzar classes Spring desde el navegador mitjançant Javascript i Prototype per a modificar la plana HTML per presentar

3.4.2. Behaviour

Aquesta llibreria permet de forma neta afegir comportament als nostres components. Aquesta incorporació es realitza mitjançant l'ús de classes CSS.

Imaginem el següent codi:

Contingut HTML

Mitjançant aquesta llibreria podem incorporar el següent codi per mostrar el contingut del tag quan clickem a sobre d'ell.

```
var myrules = {
                'b.nomClasse' : function(element){
                        element.onclick = function(){
                                alert(this.innerHTML);
                        }
                }
        };

        Behaviour.register(myrules);
```

A Canigó, s'utilitza aquesta llibreria en aquells punts en els que només amb la informació própia del DOM de l'element podem realitzar la funcionalitat. Si necessitessim de passar paràmetres a la funció es fa servir una altra alternativa (veure Ajax Tags).

Per a més referència consultar 'http://bennolan.com/behaviour/'

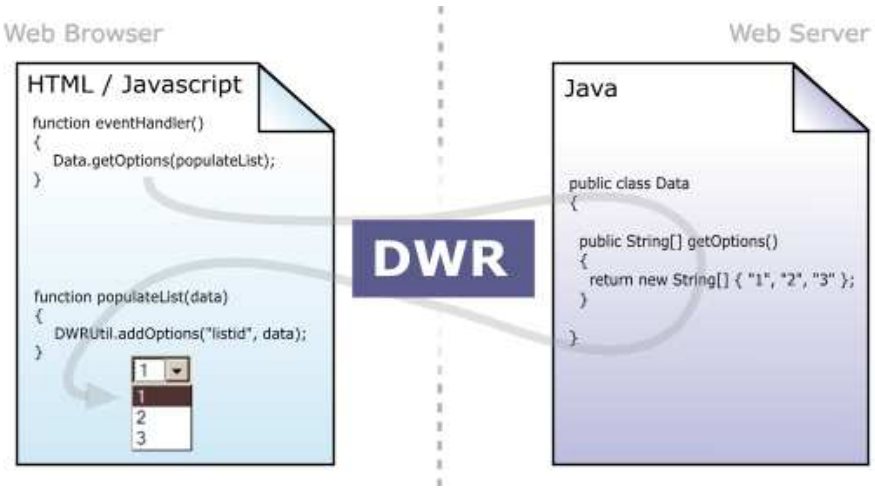
Punts de utilització a Canigó:

- Servei de Tags. Generació de codi basat en behaviour per a diferents comportaments: autoTab, selectOnFocus, trim, uppercase, lowercase, etc.

3.4.3. DWR

DWR és una llibreria basada en Ajax que permet usar des de codi Javascript del navegador funcions Java que s'executen al servidor.

La potència de DWR radica en el fet de que es genera de forma dinàmica codi Javascript a partir de classes Java. Això permet que a diferència de protocols com RMI o SOAP podem executar des del navegador funcions del servidor sense afegir cap protocol addicional a HTTP. A DWR existeix un Servlet que interpreta les crides realitzades incloent la conversió de parámetres i el retorn de valors entre Javascript i Java.



Un dels grans avantatges de DWR envers altres llibreries i el motiu d'utilització a Canigó és que DWR permet la publicació de classes Spring com a classes Java. Mitjançant això podem usar des del navegador Serveis de Canigó de forma directa.

Punts de utilització a Canigó:

- Servei de Validació. En aquest cas, mitjançant crides al servidor es poden realitzar validacions des del navegador i presentar els resultats de retorn a la mateixa plana sense necessitat de refresc
- Servei de Tags – Selects dependents d'altres selects. Refresc dinàmic dels valors d'una select quan canvii el valor de la select dependent

Una vegada publicats quins serveis mitjançant Spring volem accedir, podem accedir a la url: http://localhost:8080/nomAplicació/dwr/index.html i veure i provar les publicacions:

3.4.4. Ajax Tags

Url: http://ajaxtags.sourceforge.net/

Ajax Tags és una llibreria de tags JSP per a realitzar determinades funcionalitats basades en Ajax. Tot i que és extensa, Canigó no utilitza els tags definits per ell, però sí la llibreria javascript que fa servir internament. Aquesta llibreria està especialment indicada per crear, mitjançant l'extensió de prototype inicialitzacions als components, definició del tractament de l'event i altres funcionalitats.

Punts de utilització a Canigó:

- Servei de Tags. En tots els tags que fan ús de Ajax es fa servir la política definida per la llibreria de Ajax Tags.

3.5. Aspect Oriented Programming

Mitjançant els models de programació orientats a objectes, un canvi en els requeriments pot tenir un efecte gran en el desenvolupament. Amb Aspect Oriented Programming (AOP) podem de forma dinàmica modificar el model. Imaginem per exemple que apareix el següent requeriment: 'Degut a una petició de monitorització se'ns demana que es volqui mitjançant el Servei de Traces el temps d'execució de cada mètode de l'aplicació'. La solució, en primera instància sembla prou complicada, doncs caldria anar mètode a mètode e incorporar la crida en cada punt. A més, estaríem afegint codi a la nostra lógica de negoci que podem considerar própia de serveis adicionals.

Aquestes incorporacions són d'alguna forma requeriments secundaris doncs una classe de negoci no hauria de ser el responsable d'aquestes funcionalitats. La seva missió és realitzar la lógica de negoci.

AOP intenta resoldre aquestes problemàtiques per tal que parts que estan repartides en diferents móduls del sistema puguin ser definides en un únic punt. Aquests fragments de codi que afecten a vàries parts són els denominats aspectes i els problemes que solucionen s'anomenen 'crosscutting concerns'. Els principals punts d'aplicació AOP són:

- Rastreig i traces de les nostres execucions
- Mesura de temps i optimitzacions (profiling)
- Gestió d'excepcions
- Seguretat

Existeixen vàries implementacions AOP (Spring AOP, Nanning, AspectWerkz o AspectJ entre d'altres). La nostra recomanació és usar 'AspectWerkz', ja que permet definir mitjançant XML com realitzar la intercepció sense necessitat d'haver d'implementar cap interfície. En un punt proper es troba 'Spring AOP' tot i que aquest requerix de la implementació d'una interfície. En casos més complexos AspectJ és qui proporciona més poténcia, però la seva filosofia és força menys transparent que AspectWerkz i Spring AOP, ja que defineix una extensió al llenguatge Java i l'especificació dels aspectes es fa mitjançant Java, pel que cal una recompilació si es fan canvis.

Punts de utilització a Canigó:

- Servei d'Excepcions. Definició de la intercepció de les excepcions amb la definició de gestors per cada tipus d'excepció. Aquests poden interactuar amb altres serveis de Canigó (per exemple mitjançant el Servei de Traces emmagatzemant l'error produit)
- Servei de Seguretat. Intercepció dels objectes mitjançant l'ús de ACLs

4. Per on començar

Canigó considera tant l'arquitectura de desenvolupament com la metodologia i la gestió de la qualitat necessària per a portar a terme amb èxit un entorn d'aquesta tipologia.

Segons aquest enfoc, podem dividir els documents en 2 grups:

1) Metodologia. Conté els documents propis de la metodologia i procediments a seguir pel correcte desenvolupament de les aplicacions.
2) Framework. Conté els documents aplicables a l'arquitectura i els serveis proporcionats per Canig

4.1. Metodologia

Els documents de Metodologia estan basats en les disciplines definides amb RUP segons un enfoc àgil.

Document	Id
Metodologia de Desenvolupament i Gestió de la Qualitat Justifica l'ús de metodologies de desenvolupament en qualsevol entorn. Dóna una visió global (a alt nivell) de les diferents metodologies existents, amb els seus avantatges i desavantatges, per arribar a la meteodologia proposada en el nostre cas: RUP Agilitzat. En una segona part del document s'ofereix una introducció als aspectes de qualitat generals que es consideren en tot projecte.	MD-001
Operativa de Projectes Operativa global de projectes. Descriu les activitats i fases de la metodologia proposada, definint quins són els entregables que cal produir.	OP-001
Disciplines Operativa de Desenvolupament Operativa de l'equip de desenvolupament. Descriu el detall de les activitats a realitzar (anàlisi, disseny, construcció, proves, ...) dins l'equip per part d'analistes i programadors en el seu dia a dia, dins de les diferentes fases descrites a la metodologia.	OD-001
Entorn de Projectes Proporciona una guia d'instal·lació dels productes i eines de servidor descrites al Document d'Operativa de Projectes (cvs, maven, forge, cruise control) per implementar les pràctiques recomanades.	EP-001
Entorn de Desenvolupament Proporciona una guia d'instal·lació de l'entorn de desenvolupament local del programador, amb tots els productes i eines necessàries per treballar en la nova	ED-001

Document	Id
arquitectura segons l'operativa definida al document 'OD-001'.	
Gestió de la Qualitat Document que defineix les normatives i estàndars que s'han de seguir en els projectes basats en l'arquitectura d'execució J2EE definida. Tammateix s'ofereixen les eines necessàries per realitzar el control i seguiment del compliment de les normatives definides.	GC-001

4.2. Framework

Per cadascun dels serveis proporcionats per Canigó es proporciona un document.

Dins la Capa de Serveis Generals podem trobar-hi els següents documents:

Document	Id
Serveis de Propósit General – Servei de Validació	SG-001
Serveis de Propósit General – Servei de Traces	SG-002
Serveis de Propósit General – Servei d'Excepcions	SG-003
Serveis de Propósit General – Servei de Seguretat	SG-004
Serveis de Propósit General – Servei de Correu	SG-005
Serveis de Propósit General – Servei de Configuració	SG-006

Dins la Capa de Presentació podem trobar-hi els següents documents:

Document	Id
Serveis de Presentació – Consideracions Generals En aquest document s'ofereix una visió general de la capa de presentació, on es mostren les classes principals de l'arquitectura i el procés que es segueix per tractar qualsevol petició rebuda. Es proporciona la configuració mínima que ha de realitzar-se per tal d'usar l'arquitectura MVC bàsica de les nostres aplicacions.	SP-001
Servei de Presentació – Servei de Tags Es presenten els components d'interfície d'usuari (bàsicament formularis d'entrada de dades) i el seu ús i configuració	SP-002
Servei de Presentació – Servei Multiidioma Configuració i ús de la internacionalització a les aplicacions	SP-003
Servei de Presentació – Servei de Pantalles	SP-004
Servei de Presentació – Servei de Llistats	SP-005
Servei de Presentació – Servei XML	SP-006

Document	Id
Servei de Presentació – Servei OLE	SP-007
Servei de Presentació – Servei Upload	SP-008
Servei de Presentació – Servei Reports PDF	SP-009
Servei de Presentació – Servei Accés HTTP	SP-010

Dins la Capa d'Integració podem trobar-hi els següents documents:

Document	Id
Serveis d'Integració – Servei de Persistència	SI-001
Serveis d'Integració – Servei de FTP	SI-002
Serveis d'Integració – Servei de Cues	SI-003
Serveis d'Integració – Servei de Fitxers	SI-004
Serveis d'Integració – Servei de Planificació de Tasques	SI-005
Serveis d'Integració – Servei d'Integració amb WebServices	SI-006

Dins els Connectors Funcionals podem trobar-hi els següents documents:

Document	Id
Connectors Funcionals – Connector CatCert	CF-001
Connectors Funcionals – Connector Documentum	CF-002
Connectors Funcionals – Connector GECAT	CF-003
Connectors Funcionals – Connector S@rcat	CF-004