



---

## **Framework Corporatiu J2EE**

### **Preguntes Freqüents**

**Versió 1.2**

Barcelona, 15 / març / 2007



## Històric de modificacions

Data	Autor	Comentaris	Versió
21/03/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0
16/06/2006	Atos Origin, sae openTrends	Afegides consultes/incidències del suport a Justícia	1.1
15/03/2007	Atos Origin, SAE	Actualització del document	1.2

### Llegenda de Marcadors



## Índex

<b>1. PREGUNTES COMUNS</b>	<b>4</b>
1.1. ERRORS EN ARRENCAR L'APLICACIÓ	4
1.1.1. <i>Error creating bean with name 'sessionFactory'</i>	4
1.1.2. <i>Bean context must contain at least one bean of type net.sf.acegisecurity.util.FilterChainProxy</i>	6
1.1.3. <i>Apareix el missatge 'Usuari o password incorrecte' però crec que està tot correctament configurat</i>	7
1.1.4. <i>Apareix el missatge 'ClassCastException... TilesRequestProcesso.initDefinitionsMapping'</i>	8
1.2. PROBLEMES EN ECLIPSE	9
1.2.1. <i>Obrir un fitxer tarda moltíssim i es queda el sistema com aturat</i>	9
1.2.2. <i>On trobo les traces del Servidor Tomcat?</i>	10
1.2.3. <i>He fet canvis a un fitxer però sembla que no es reflecteixen en l'execució</i>	11
1.2.4. <i>Unsupported major.minor version 49.0</i>	12
1.3. PROBLEMES AMB MAVEN A ECLIPSE	15
1.3.1. <i>Quan es descarreguen les dependències de llibreries dona ConnectionTimeout</i>	15
1.4. CONSULTES / INCIDÈNCIES DIVERSES	16
1.4.1. <i>En l'aplicació les modificacions de la BD les fem en un ordre i en canvi per la consola es veuen en ordre invers, provocant un error a la BD del tipus DataIntegrityViolationException</i>	16
1.4.2. <i>Com podem fer servir sentències SQL en lloc de HQL als llistats de l'aplicació, si per a generar els llistats fem servir el servei de llistats?</i>	17
1.4.3. <i>Com puc implementar la cerca aproximada mitjançant LIKE a les consultes associades als llistats? Com puc fer cerques case insensitive?</i>	20
1.4.4. <i>Com podem canviar els controls de navegació en els llistats per tal de que en lloc d'imatges siguin botons?</i>	21
1.4.5. <i>Com puc mostrar més de un llistat en una pàgina fent servir el servei de llistats?</i>	24
1.4.6. <i>Com puc generar un llistat amb el servei de llistats on la consulta HQL estigui formada per una JOIN i per pantalla es mostri informació de tots els objectes que conformen la JOIN?</i>	27
1.4.7. <i>L'exportació dels llistats a PDF i Excel només m'exporta la primera pàgina</i>	29
1.4.8. <i>Com podem modificar els paràmetres de cerca que rep el servei de llistats de la Request?</i>	30
1.4.9. <i>Com podem afegir nous paràmetres de cerca per a les consultes que retornen les dades de les combos?</i>	31
1.4.10. <i>Com puc implementar el meu propi custom editor?</i>	33
1.4.11. <i>Com puc generar un desplegable amb les dades d'una taula mestre?</i>	36
1.4.12. <i>Error amb les connexions amb Tomcat: Socket closed</i>	38
1.4.13. <i>Exemple d'invocació client-servidor mitjançant AJAX</i>	39
1.4.14. <i>Com modificar el comportament de la gestió d'errors que té DWR?</i>	47
1.4.15. <i>Com puc afegir un CustomEditor per les dades de tipus Number?</i>	48
1.4.16. <i>Problemes amb els elements enllaçats o depenents</i>	49
1.4.17. <i>Més d'un formulari en una mateixa JSP</i>	51
1.4.18. <i>Eliminar col.lumnes d'un llistat a l'exportar a excel o pdf</i>	55
1.4.19. <i>Més d'una ValueList en una mateixa JSP</i>	57



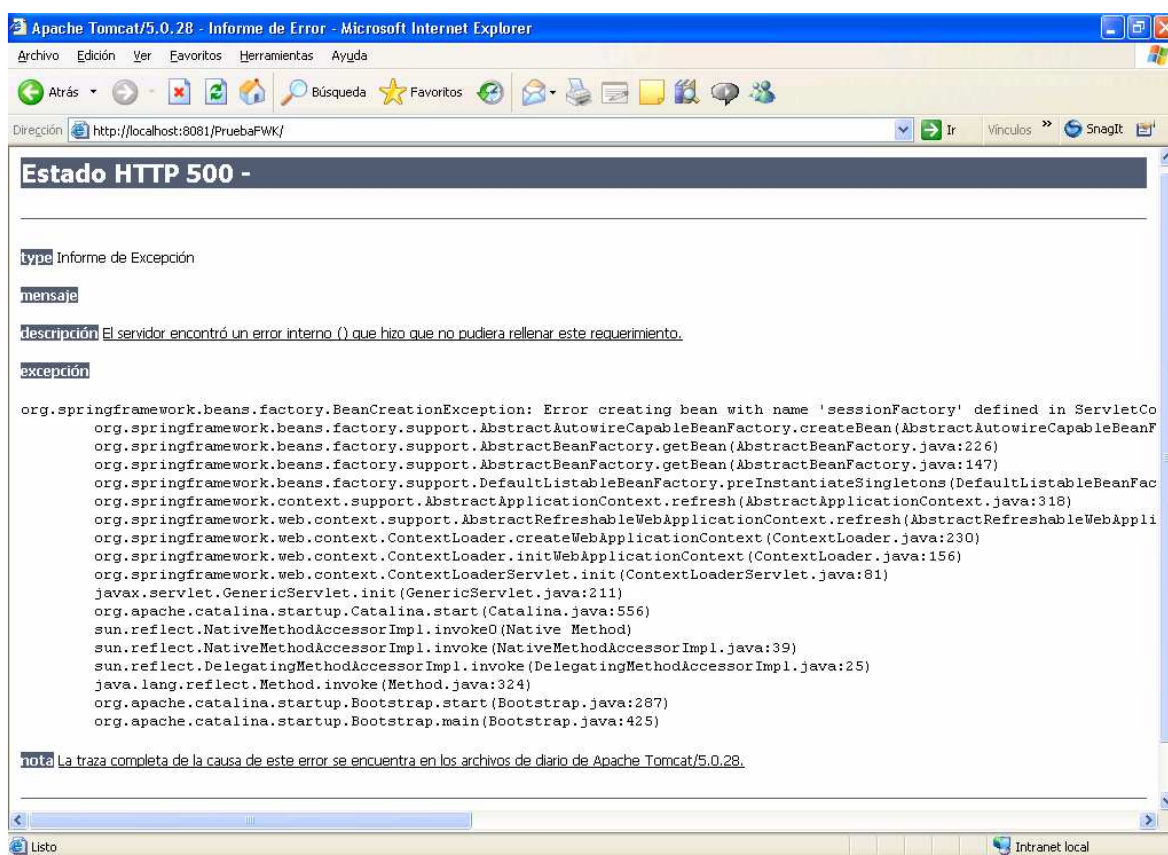
## 1. Preguntes Comuns

### 1.1. Errors en Arrencar l'Aplicació

#### 1.1.1. Error creating bean with name 'sessionFactory'...

##### Problema

Si apareix un missatge com el mostrat a continuació:



##### Solució

En general, aquest problema és degut a que no s'ha definit correctament el datasource al Servidor. Cada servidor té la seva forma particular de definir el datasource.

En el cas de fer ús de Tomcat, podem incorporar al fitxer 'server.xml' el següent codi:

Dins el tag 'Context' de l'aplicació afegir un 'Resource' per definir la connexió a la font de dades.



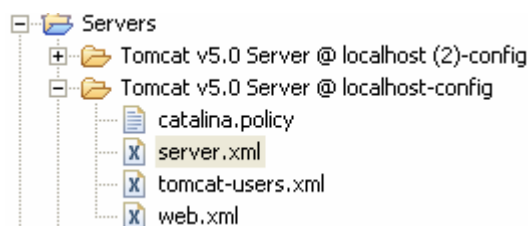
Exemple:

```
<Context docBase="D:\Users\xescuder\Projectes\openFrame-samples-
jPetstore\deployables\openFrame-samples-jPetstore" path="/openFrame-samples-
jPetstore" reloadable="true">

<Resource name="openFrameDS" auth="Container" type="javax.sql.DataSource"/>
  <ResourceParams name="openFrameDS">
    <parameter>
      <name>driverClassName</name>
      <value>oracle.jdbc.driver.OracleDriver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:oracle:thin:@10.32.14.21:1521:tsydes9</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>gpp_adm</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>gpp_adm</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>20</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>10</value>
    </parameter>
    <parameter>
      <name>maxWait</name>
      <value>-1</value>
    </parameter>
  </ResourceParams>

</Context>
```

En Eclipse podem accedir a aquest fitxer des de la carpeta 'Servers>Nom Servidor'.





### **1.1.2. Bean context must contain at least one bean of type *net.sf.acegisecurity.util.FilterChainProxy***

#### **Problema**

```
javax.servlet.ServletException: Bean context must contain at least one bean of
type net.sf.acegisecurity.util.FilterChainProxy
    net.sf.acegisecurity.util.FilterToBeanProxy.doInit(FilterToBeanProxy.java:200)
    net.sf.acegisecurity.util.FilterToBeanProxy.doFilter(FilterToBeanProxy.java:122
)

Si fas F5 canviarà el missatge a:

java.lang.NullPointerException
    net.sf.acegisecurity.util.FilterToBeanProxy.doFilter(FilterToBeanProxy.java:125
)
```

#### **Solució**

Aquest missatge apareix perquè s'ha definit el filtre del Servei de Seguretat de la classe 'net.sf.acegisecurity.util.FilterToBeanProxy'.

```
<filter>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <filter-class>
    net.sf.acegisecurity.util.FilterToBeanProxy
  </filter-class>
```

En cas de que no es vulgui activar la seguretat comentar aquest filtre i el seu filter-mapping associat.

Si es vol activar la seguretat realment ens hem d'assegurar que:

- S'ha definit la dependència amb la llibreria 'openFrame-services-security'
- S'ha incorporat la referència al fitxer 'acegi-beans.xml' que està contingut en l'anterior llibreria

El Servei de Seguretat defineix alguns beans de configuració dins el jar desplegat. Ara bé, per tal que aquests siguin carregats hem de definir la seva importació en els nostres fitxers de configuració:

```
<import resource="classpath:/spring/acegi-beans.xml" />
```



### 1.1.3. Apareix el missatge 'Usuari o password incorrecte' però crec que està tot correctament configurat

#### Problema

El presentat en el títol.

#### Solució

Aquest problema es pot donar si no s'han incorporat els fitxers de càrrega de hibernate necessaris pel Servei de Seguretat.

Seguir els següents pasos:

- 1) Accedir al fitxer 'openFrame-services-configuration.xml' i observar quin és el fitxer 'jdbc.properties' que es fa servir (aquest hauria de ser diferent per cada tipus de Servidor: WebSphere, WebLogic, Tomcat, ...)

```
<beans>
  <bean id="configurationService"
    class="net.opentrends.openframe.services.configuration.springframework.beans.factory.config.HostPropertyPlaceholderConfigurer">
    <property name="basePropertyFiles">
      <list>
        <value>classpath:jdbc/jdbc.properties</value>
        <value>classpath:mail/mail.properties</value>

        <value>classpath:file/fileUploadService.properties</value>
        <value>classpath:file/fileService.properties</value>
      </list>
    </property>
  </bean>
</beans>
```

- 2) Al fitxer corresponent ('jdbc.properties', 'jdbc.websphere.properties', etc.) mirar quin és el valor de la variable 'sessionFactory.configLocation'

```
sessionFactory.configLocation=hibernate/config/hibernate.cfg.xml
```

- 3) Obrir el fitxer referenciat al valor del paràmetre i assegurar-se que en l'apartat de mappings apareixen les següents referències:

```
<mapping resource="hibernate/mappings/UserLogin.hbm.xml" />
<mapping resource="hibernate/mappings/PartyGroup.hbm.xml" />
<mapping resource="hibernate/mappings/Role.hbm.xml" />
```

Aquestes referències són necessàries pel Servei de Seguretat. Els fitxers es troben dins el jar de 'openFrame-services-security'.



#### 1.1.4. Apareix el missatge 'ClassCastException... TilesRequestProcessor.initDefinitionsMapping'

##### Problema

```
java.lang.ClassCastException  
    org.apache.struts.tiles.TilesRequestProcessor.initDefinitionsMapping(TilesRequestProcessor.java:84)
```

##### Solució

Cal definir el plugin de Tiles i la ruta a la definició de les pantalles:

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">  
  <set-property property="definitions-config" value="/WEB-INF/classes/tiles/tiles-definitions.xml"/>  
  <set-property property="moduleAware" value="true" />  
</plug-in>
```



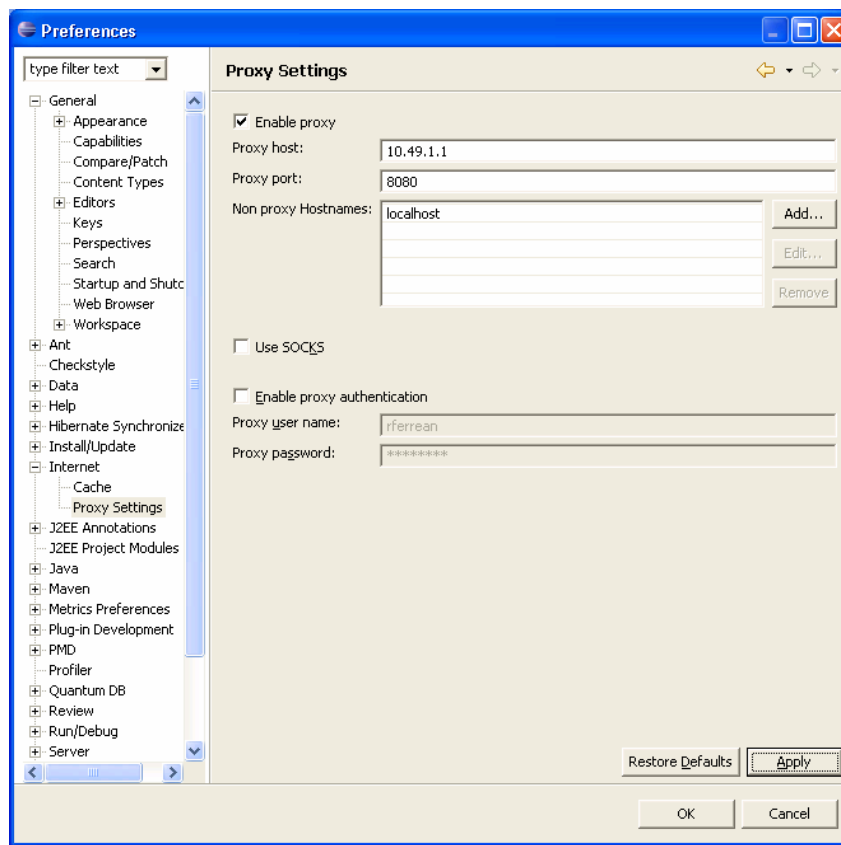
## 1.2. Problemes en Eclipse

### 1.2.1. Obrir un fitxer tarda moltíssim i es queda el sistema com aturat

Aquest comportament és degut a la configuració del proxy que valida els fitxers.

Accedir a l'opció 'Window>Preferences' i seleccionar el node 'Internet>Proxy Settings'.

Si per accedir a la Web es fa ús d'un Proxy configurarem els seus valors, en cas contrari desmarcar 'Enable proxy':





### **1.2.2. On trobo les traces del Servidor Tomcat?**

Podem accedir a les traces d'arranc de Tomcat per veure si hi ha algun problema a la carpeta '\.metadata\plugins\org.eclipse.wst.server.core' del workspace utilitzat.

Dins aquesta carpeta escollir el subdirectori 'tmpX' corresponent al servidor i accedir al directori 'logs'.



### ***1.2.3. He fet canvis a un fitxer però sembla que no es reflecteixen en l'execució***

El directori '.deployables' és el directori que utilitza Tomcat en els desplegaments. Si hem fet un canvi a un fitxer dins l'estructura arrel de src, primer de tot ens assegurarem que s'ha copiat correctament al subdirectori '.deployables'.

Si no s'ha copiat a '.deployables' hem d'assegurar-nos de que al fitxer .classpath (a l'arrel del projecte) tenim el següent contingut:

```
<classpathentry output=".deployables/WEB_NAME/WEB-INF/classes" kind="src"
path="src/main/java"/>
  <classpathentry output=".deployables/WEB_NAME/WEB-INF/classes"
kind="src" path="src/main/resources"/>
```

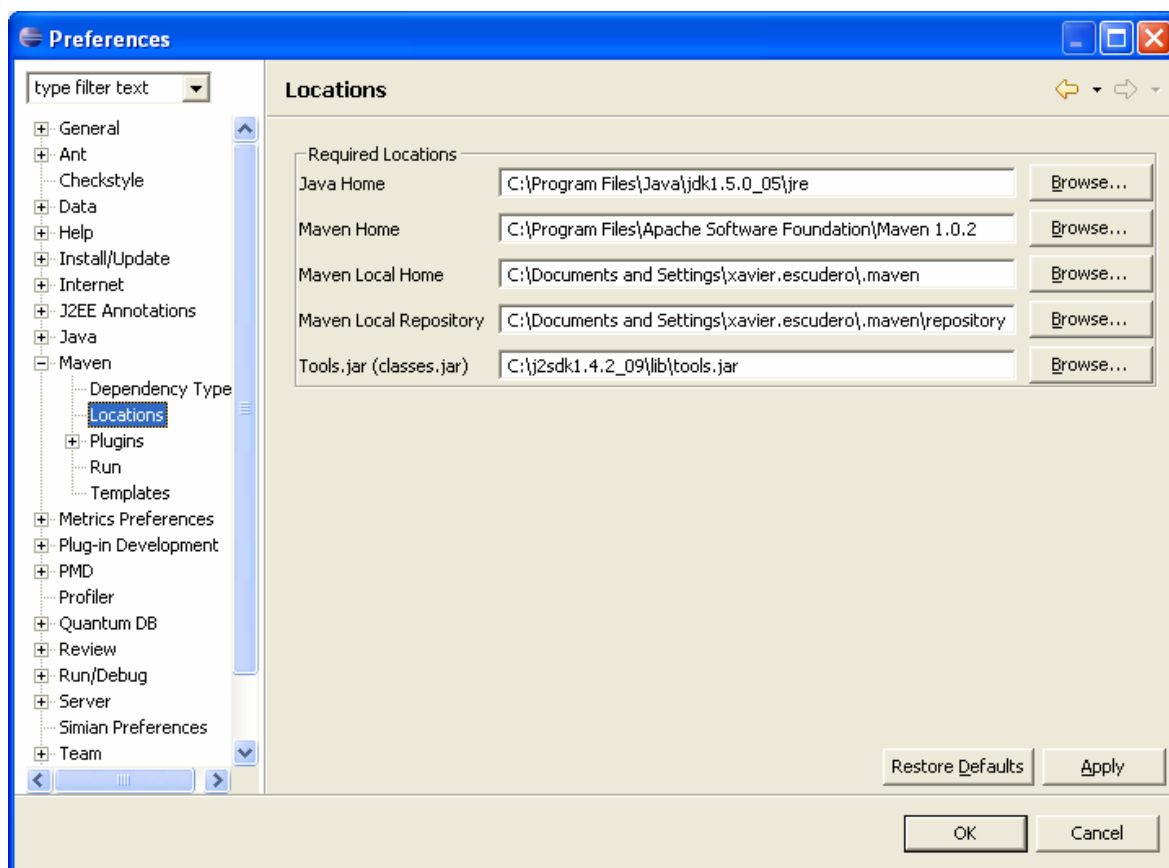
On WEB\_NAME correspon al subdirectori dins de deployables que correspon a l'aplicació.

### 1.2.4. *Unsupported major.minor version 49.0*

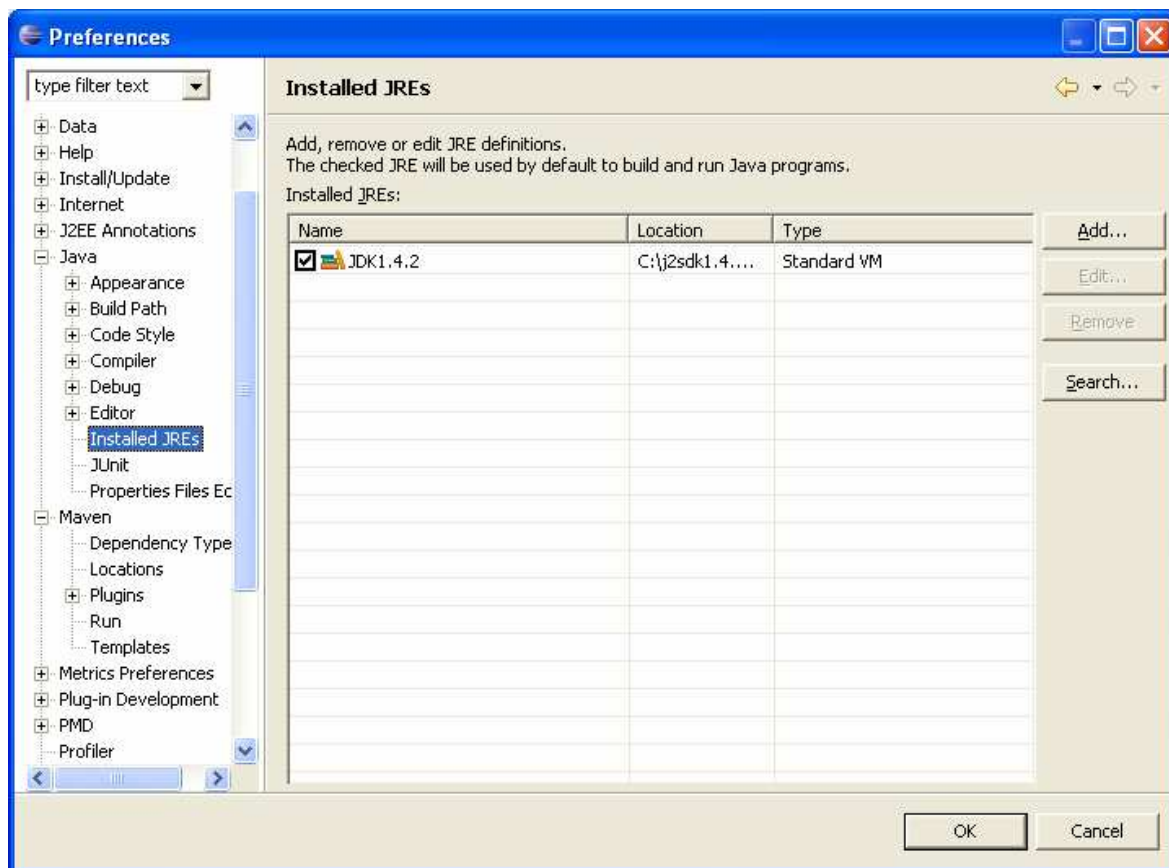
```
BUILD FAILED
File..... C:\Documents and Settings\xavier.escudero\.maven\cache\maven-
java-plugin-1.5\plugin.jelly
Element... ant:javac
Line..... 63
Column.... 48
com/sun/tools/javac/Main (Unsupported major.minor version 49.0)
```

Window>Preferences.

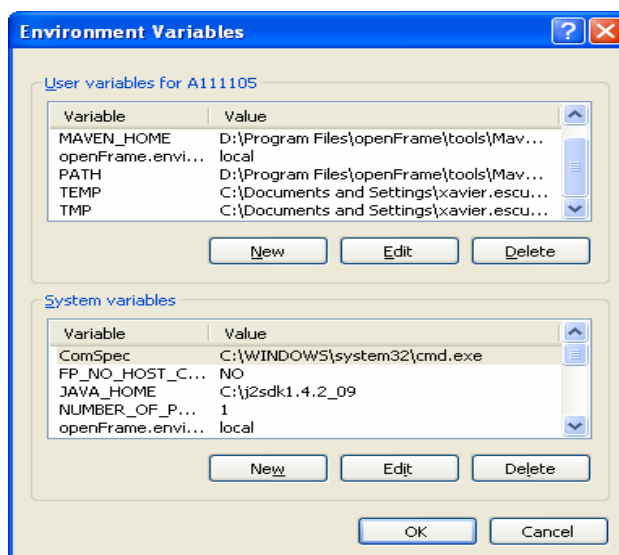
Canviar 'Java Home' per a que apunti al j2sdk1.4.2\_09.

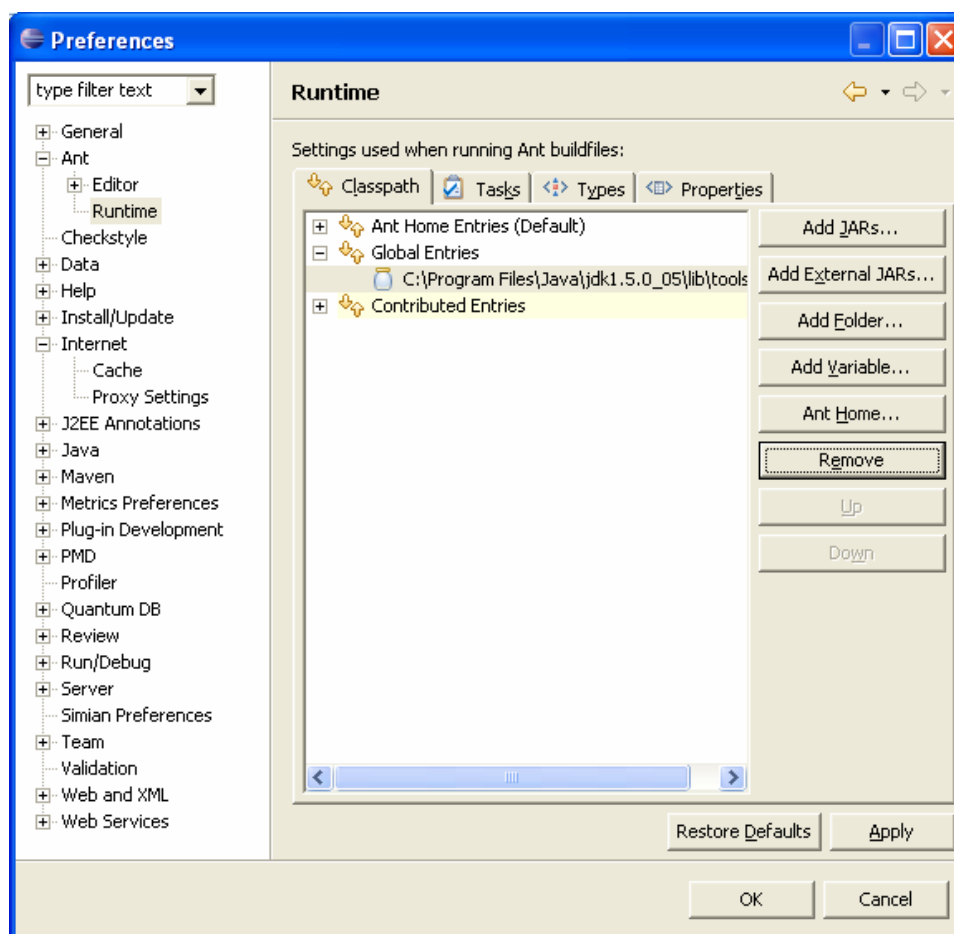


Assegurar-se al node 'Java>Installed JREs' que tenim seleccionat per defecte 'JDK1.4.2'.



Des del Panel de Control>Sistema assegurar-se que la variable d'entorn apunta al JSDK1.4.2







## 1.3. Problemes amb Maven a Eclipse

### 1.3.1. Quan es descarreguen les dependències de llibreries dóna *ConnectionTimeout*

#### Problema

En executar algun goal per descarregar les dependències definides al fitxer 'project.xml' dóna problemes de connexió o es queda bastant temps aturat.

La seva resolució normalment depèn de la xarxa utilitzada però en la majoria de casos esdevé mitjançant la configuració del proxy a Internet.

#### Solució

En cas de que es faci servir proxy de connexió a Internet, podem definir que es faci ús del proxy amb els següents passos:

- 1) Crear un fitxer amb nom 'build.properties' i ubicar-lo al directori 'C:/Documents and Settings/<username>'. Aquest fitxer ha de ser definit per cada desenvolupador.
- 2) En aquest fitxer definir les variables 'maven.proxy.host' i 'maven.proxy.port'.

Exemple :

```
maven.proxy.host=10.49.1.1  
maven.proxy.port=8080
```

Si una vegada aplicats els canvis apareix un missatge com el mostrat a continuació:

```
Intentando descargar jdtcore-3.1.0.jar.  
No credentials available for the 'null' authentication realm at  
proxy.intranet.atosorigin  
Error retrieving artifact from  
[http://www.opentrends.net/repository/eclipse/jars/jdtcore-3.1.0.jar]:  
java.io.IOException: Not authorized by proxy.
```

Cal afegir les variables 'maven.proxy.username' i 'maven.proxy.password'.



## 1.4. Consultes / incidències diverses

### 1.4.1. *En l'aplicació les modificacions de la BD les fem en un ordre i en canvi per la consola es veuen en ordre invers, provocant un error a la BD del tipus `DataIntegrityViolationException`.*

#### Problema

El descrit a la pregunta

#### Solució

Aquest error es deu a l'ordre en el qual hibernate persisteix els canvis realitzats en els objectes persistents sobre la BD.

Podem trobar informació sobre aquest procediment al següent link:

[http://www.hibernate.org/hib\\_docs/v3/reference/en/html\\_single/#objectstate-flushing](http://www.hibernate.org/hib_docs/v3/reference/en/html_single/#objectstate-flushing)

Per exemple, si tenim el següent codi dintre d'un mètode d'un DAO:

```
// account és l'objecte que volem eliminar i tornar a guardar.  
super.delete(account.getId()); (1)  
...  
super.save(account); (2)
```

provocarà una excepció del tipus [org.springframework.dao.DataIntegrityViolationException](#) quan finalitzi la transacció.

Això es deu a que l'ordre en el qual es persisteixen els canvis és el següent (està més detallat al link anterior):

- Primer tots els inserts de entitats. (pas 2 de l'exemple)
- ...
- Per últim, tots els delete d'entitats. (pas 1)

La solució a aquest problema consisteix en forçar un *flush* després de fer el *delete*. És a dir:

```
// account és l'objecte que volem eliminar i tornar a guardar.  
super.delete(account.getId());  
super.getSession().flush();  
...  
super.save(account);
```

**Important:** no cal confondre el *flush* amb un *commit*. El *flush* no fa *commit*, simplement volca els canvis sobre la BD però no els fa persistents. Per tant un *rollback* donaria marxa enrera als canvis.





### 1.4.2. Com podem fer servir sentències SQL en lloc de HQL als llistats de l'aplicació, si per a generar els llistats fem servir el servei de llistats?

#### Problema

Volem fer servir sentències SQL per a obtenir les dades pels llistats generats via la llibreria Valuelist.

#### Solució

Sabem que el servei de llistats de l'openFrame es fonamenta en la llibreria open source Valuelist (<http://valuelist.sourceforge.net>).

Els motius que podem fer-nos fer servir SQL en lloc de HQL són molt diversos:

- Complexitat alhora de traduir la sentència SQL a HQL.
- Ús de elements de la BD que no tenen un pojo associat (podem ser taules temporals, vistes, taules que no són de negoci, ...).
- ...

En el cas que volguem fer servir SQL en lloc de HQL, l'únic canvi que hem de fer és a nivell de configuració, ja que la JSP no sap en cap moment de quina font provenen les dades que està renderitzant.

Els canvis a fer són els següents:

- Modificar el fitxer openFrame-services-web-list.xml, per tal de definir la consulta SQL amb el seu adaptador.
- Definir un wrapper que permeti fer el mapeig de les dades retornades per la consulta a objectes Java.

#### Modificació del fitxer openFrame-services-web-list.xml

La modificació a fer en aquest fitxer consisteix a definir una entrada dintre de l'apartat de *config.adapters* corresponent a la nostra consulta. L'única diferència important respecte a la resta d'entrades és l'adapter que es fa servir, que no correspon a l'adapter *baseHibernateAdapter* típic de les consultes HQL sinó a un nou adapter.

Aquí podem veure un exemple:

```
<entry key="accountListSQL">
  <bean class="net.mlw.vlh.adapter.jdbc.objectWrapper.DefaultWrapperAdapter">
    <property name="dataSource"><ref bean="dataSourceSQL"/></property>
    <property name="defaultNumberPerPage"><value>4</value></property>
    <property name="defaultSortColumn"><value>userid</value></property>
    <property name="defaultSortDirection"><value>asc</value></property>
    <property name="sql">
      <value>
        SELECT vo.*, category.*
        FROM
        Account vo, Category category
        WHERE 1=1 and vo.preferred_category = category.CATID
        /~firstname: AND lower(vo.firstname) LIKE lower('[firstname]') ~/
        /~lastname: AND lower(vo.lastname) LIKE lower('[lastname]') ~/
        /~preferredCategory.id: AND vo.preferred_Category = {preferredCategory.id} ~/
        /~lower18: AND vo.is_lower18 = {lower18} ~/
        /~sortColumn: ORDER BY vo.[sortColumn] [sortDirection]~/
      </value>
    </property>
    <property name="wrapper"><ref bean="accountWrapper"/></property>
  </bean>
</entry>
```



```
<property name="wrapResultSet"><value>true</value></property>
</bean>
</entry>
```

Com podem veure, definim un adaptador de tipus *DefaultWrapperAdapter*, al qual li hem d'indicar quin *dataSource* farà servir per a obtenir les connexions a la BD, li configurem els paràmetres típics de tot llistat: nombre d'elements per pàgina, columna d'ordenació per defecte, ..., li indiquem la consulta SQL i per últim, **li indiquem quina classe ha de fer servir per a fer el mapeig del resultset a objectes Java.**

En aquest cas li hem indicat una referència a un bean de nom *accountWrapper*, la definició del qual és la següent (la podem ficar dintre del mateix fitxer de configuració):

```
<bean id="accountWrapper"
class="net.opentrends.openframe.formacio.model.jdbc.wrapper.AccountWrapper"/>
```

Respecte a la referència al *dataSource*, en el fitxer *openFrame-services-persistence.xml* podem tenir definit el següent bean (mateix funcionament que un look up mitjançant un *ServiceLocator*):

```
<bean id="dataSourceSQL" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName"><value>${dataSource.jndiName}</value></property>
</bean>
```

### **Definició de la classe wrapper**

Hem de definir un mecanisme que ens permeti llegir les dades del resultset i encapsular-les en un objecte Java per tal de que siguin afegides al llistat del value list.

A continuació podem veure la implementació de *AccountWrapper*: en aquest cas el que fem és traspasar les dades del resultset a un objecte *Account* (el mateix pojo que tenim definit al *petStore*). Podem optar per aixecar completament el pojo o només informar aquelles columnes a les quals es fa referència al llistat. En aquest exemple s'ha optat per la segona alternativa, però és recomanable optar per la primera ja que futures modificacions sobre la presentació del llistat (per exemple afegir més informació) no implicarien modificar el wrapper.

```
public class AccountWrapper implements ObjectWrapper {

    public AccountWrapper() {
        super();
    }

    public Object getWrappedRecord(Object arg0) {
        /*
         * Aquest mètode pot rebre o bé tot l'objecte resultSet.next o bé
         * una columna específica de l'objecte. Això es defineix en la configuració
         * de l'adapter mitjançant la propietat wrapResultSet.
         *
         * El mètode ha de retornar l'objecte bean que s'acabarà afegint a la
         * llista que retorna el valueList.
         */
        ResultSet res = (ResultSet)arg0;
```



```
/* Objecte a retornar. L'aixequem completament, per la qual cosa ens caldrà generar el  
seu id. */  
Account wObject = null;  
  
try{  
  
    Category wCategory = null;  
  
    /* Aixequem l'objecte wCategory. */  
    String sCatId = res.getString(19);  
    String sName = res.getString(20);  
    String sDescn = res.getString(21);  
    wCategory = new Category(sCatId);  
    wCategory.setName(sName);  
    wCategory.setDescn(sDescn);  
  
    /* Aixequem l'objecte wAccount. */  
    String sUserId = res.getString(1);  
    String sFirstName = res.getString(3);  
    String sLastName = res.getString(4);  
  
    wObject = new Account (sUserId, wCategory, null, null,null,null, null,sFirstName,  
sLastName, null, null, null, null);  
  
    }catch(Exception e){e.printStackTrace();}  
  
    return wObject;  
}  
  
public void setValueListInfo(ValueListInfo arg0) {  
    // No implementat en aquest exemple.  
}  
}
```

Fixem-nos que el nostre wrapper ha d'implementar la interfície *net.mlw.vlh.adapter.util.ObjectWrapper*.

Com ja s'ha comentat, el mètode *getWrappedRecord* ha de retornar l'objecte java que serà afegit al llistat del value list. Aquest mètode rep com a paràmetre el registre del Resultset a llegir, per tant **no hem de fer cap iteració** sobre aquest.

Per últim, el fet de fer servir la posició de la columna en lloc del nom de la columna és merament anecdòtic. Es pot fer servir qualsevol de les dos alternatives.

Podem trobar més informació a la següent adreça:

<http://valuelist.sourceforge.net/adapters/DefaultWrapperAdapter.html>



### **1.4.3. Com puc implementar la cerca aproximada mitjançant LIKE a les consultes associades als llistats? Com puc fer cerques case insensitive?**

#### **Problema**

Volem implementar a les nostres consultes dels llistats la funcionalitat de cerca LIKE i case insensitive de SQL.

#### **Solució**

A continuació podem veure un exemple en el qual es fa la cerca aproximada respecte a l'atribut *firstname* del pojo *account*. En aquest exemple també podem veure un exemple de cerca aproximada i sense considerar majúscules i minúscules sobre l'atribut *lastname*.

```
<property name="hql">
  <value>
    FROM
      net.opentrends.openframe.formacio.model.Account AS vo
    WHERE 1=1
    /~firstname: AND vo.firstname LIKE ('[firstname]') ~/
    /~lastname: AND lower(vo.lastname) LIKE lower('%[lastname]') ~/
    /~sortColumn: ORDER BY vo.[sortColumn] [sortDirection]~/
  </value>
</property>
```

#### 1.4.4. Com podem canviar els controls de navegació en els llistats per tal de que en lloc d'imatges siguin botons?

##### Problema

El descrit a la pregunta.

##### Solució

La configuració d'aquestes imatges la trobem definida als fitxers de recursos de l'aplicació (generalment els tindrem dintre del directori resources/i18n).

En aquests fitxers tenim definides les següents entrades:

```
paging.first(off)=
paging.first(on)=
paging.previous(off)=
paging.previous(on)=
paging.forward(off)=
paging.forward(on)=
paging.last(off)=
paging.last(on)=

paging.focus(on)=
paging.focus(off)=
paging.focus(error)=
paging.focus(disabled)=

paging.text.totalRow={0} Total,
paging.text.pageFromTotal= <b>{0}</b> of {1} page(s)
```

Aquestes definicions són les que fa servir el value list per a generar tota la part de paginació (dintre del fitxer *valueList.tld*, si cerquem l'entrada *showSummary* podem trobar quina és l'estructura del codi de la paginació).

Per tant, si volem modificar les imatges que fem servir per a navegar pel llistat només haurem de modificar les definicions anteriors.

L si en lloc d'imatges volem fer servir botons el que s'ha de fer és canviar el codi HTML associat al fitxer de recurs per a cada entrada (en lloc del tag `img` fer servir el tag `input`, per exemple). Per exemple, si definim el següent codi al fitxer de recursos:

```

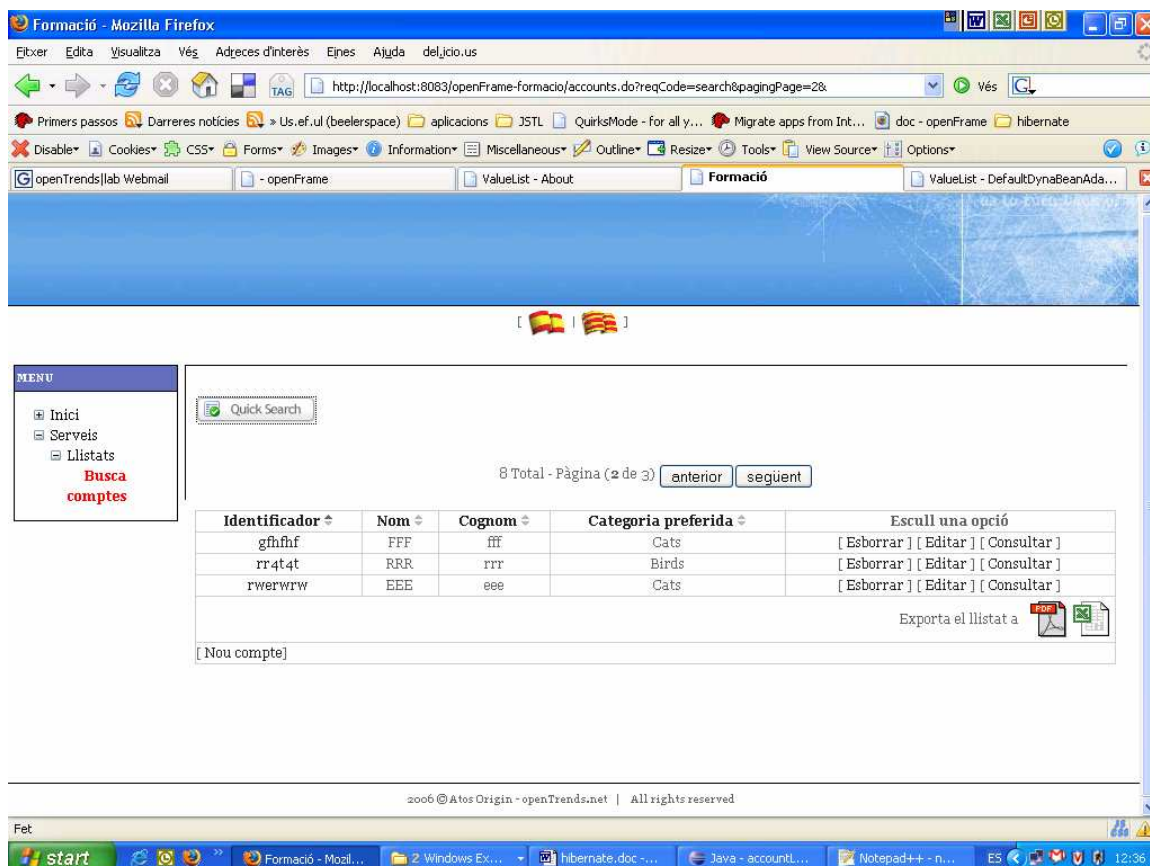
paging.delim=
paging.first(off)=
paging.first(on)=
paging.previous(off)=
paging.previous(on)=<input type=submit value="anterior" />
paging.forward(off)=
paging.forward(on)=<input type=submit value="següent" />
paging.last(off)=
paging.last(on)=

paging.focus(on)=
paging.focus(off)=
paging.focus(error)=
paging.focus(disabled)=

paging.text.totalRow={0} Total -
paging.text.pageFromTotal= Pàgina (<b>{0}</b> de {1})

```

Obtenim





Es recomana mirar-se el fitxer *valuelist.tld* per a veure més propietats de la generació de llistats mitjançant value list.



### 1.4.5. Com puc mostrar més de un llistat en una pàgina fent servir el servei de llistats?

#### Problema

El descrit a la pregunta

#### Solució

Anem a mostrar un exemple en el qual es mostren per pantalla dos llistats: un de comptes i un altre de categories.

Per a cadascun dels llistats limitarem la seva secció mitjançant el tag  
`<vlh:root>...</vlh:root>`

i per a distingir cadascun d'aquests haurem de definir un identificador per a cadascun d'ells:  
`<vlh:root id="****">...</vlh:root>`

Aquest identificador es fa servir per a identificar a quin llistat pertany els camps de paginació o ordenació.

La clau del procés està en fer servir el tag `<vlh:retrieve name="id_de_consulta">` que ens permet associar a cadascun dels llistats una consulta definida en el fitxer de configuració.

El següent codi JSP (només ens mostra la part realment important):

```
<%@ include file="/WEB-INF/jsp/includes/fwkJtagLibs.jsp" %>
&nbsp;<br>
<!-- Primer llistat -->
<vlh:root id="(comptes)" value="list" url="accounts.do?"
includeParameters="reqCode" configName="vlConfig">
  <vlh:retrieve name="accountList">
    <vlh:attribute name="pagingNumberPer" value="3"/>
  </vlh:retrieve>
  <table width="700" align="center" border="0">
    <!-- pagination section -->
    <tr>
      <vlh:paging showSummary="true" attributes="align=right"/>
    </tr>
    <!-- end of Pagination -->

    <vlh:row bean="account" display="<%=displayProvider%>">
      <vlh:column titleKey="jsp.accounts.accountList.id"
property="id" sortable="desc">
        ...
      </vlh:row>

  </table>
</vlh:root>

<br><br>
```





```
<!-- Segon llistat -->
<vlh:root id="(categories)" value="list" url="accounts.do?"
includeParameters="reqCode" configName="vlConfig">
  <vlh:retrieve name="categoryList">
    <vlh:attribute name="pagingNumberPer" value="4"/>
  </vlh:retrieve>
  <table width="700" align="center" border="0">
    <!-- pagination section -->
    <tr>
      <vlh:paging showSummary="true" attributes="align=right"/>
    </tr>
    <!-- end of Pagination -->
    <vlh:row bean="category" >
      <vlh:column titleKey="jsp.categories.category.id"
property="id" sortable="desc"/>
      ...
    </vlh:row>
  </table>
</vlh:root>
```



ens genera la següent sortida per pantalla:

Formació - Mozilla Firefox

http://localhost:8083/openFrame-formacio/accounts.do?reqCode=search

8 Total - Pàgina (1 de 3) [següent](#)

Identificador	Nom	Cognom	Categoria preferida	Escull una opció
322	EEEE	eee	Cats	[ Esborrar ] [ Editar ] [ Consultar ]
3242	ERET	tertet	Cats	[ Esborrar ] [ Editar ] [ Consultar ]
7878	RRRRR	67868	Cats	[ Esborrar ] [ Editar ] [ Consultar ]

Exporta el llistat a  

[ Nou compte ]

5 Total - Pàgina (1 de 2) [següent](#)

Identificador	Nom	Descripció
BIRDS	Birds	Birds
CATS	Cats	Cats
DOGS	Dogs	Dogs
FISH	Fish	Fish



Per a aquest exemple s'ha fet servir butons per a la paginació en lloc d'imatges (tal i com s'ha explicat en una pregunta anterior).

### **Important**

Aquest mecanisme de generació de llistats té el handicap de que no fa servir els paràmetres que estiguin a la sessió, i per tant no té '*memòria*' de l'estat anterior.



#### 1.4.6. Com puc generar un llistat amb el servei de llistats on la consulta HQL estigui formada per una JOIN i per pantalla es mostri informació de tots els objectes que conformen la JOIN?

##### Problema

El descrit a la pregunta.

##### Solució

Anem a veure un exemple de consulta HQL amb una JOIN, i com indicar a la JSP la informació que volem mostrar.

Per a fer servir aquesta funcionalitat el primer que cal fer **és actualitzar-se les llibreries del framework** *openFrame-services-core* i *openFrame-services-web* a la versió 1.0.1-SNAPSHOT.

Per a veure l'exemple, agafem la següent consulta HQL (la qual tenim definida dintre del fitxer *openFrame-services-web-list.xml*):

```
<entry key="accountList">
  <bean parent="baseHibernateAdapter">
    <property name="hql">
      <value>
        SELECT vo, pr
        FROM
        net.opentrends.openframe.formacio.model.Account AS vo,
        net.opentrends.openframe.formacio.model.Product AS pr
        WHERE 1=1 AND vo.preferredProduct.id = pr.id
        /~firstname: AND lower(vo.firstname) LIKE lower('[firstname]') ~/
        /~lastname: AND lower(vo.lastname) LIKE lower('[lastname]') ~/
        /~preferredCategory.id:      AND      vo.preferredCategory.id      =
        {preferredCategory.id} ~/
        /~lower18: AND vo.lower18 = {lower18} ~/
        /~sortColumn: ORDER BY vo.[sortColumn] [sortDirection]~/
      </value>
    </property>
  </bean>
</entry>
```

Podem veure que tenim definida una JOIN entre Account i Product.

En la pàgina JSP l'únic que hem de fer per a recuperar informació d'un objecte o de l'altre és fer servir la següent sintaxis:

*nom\_del\_bean.array[posicio].propietat*

on

*nom\_del\_bean*: correspon al nom del bean amb el qual treballa el <vlh:row>.

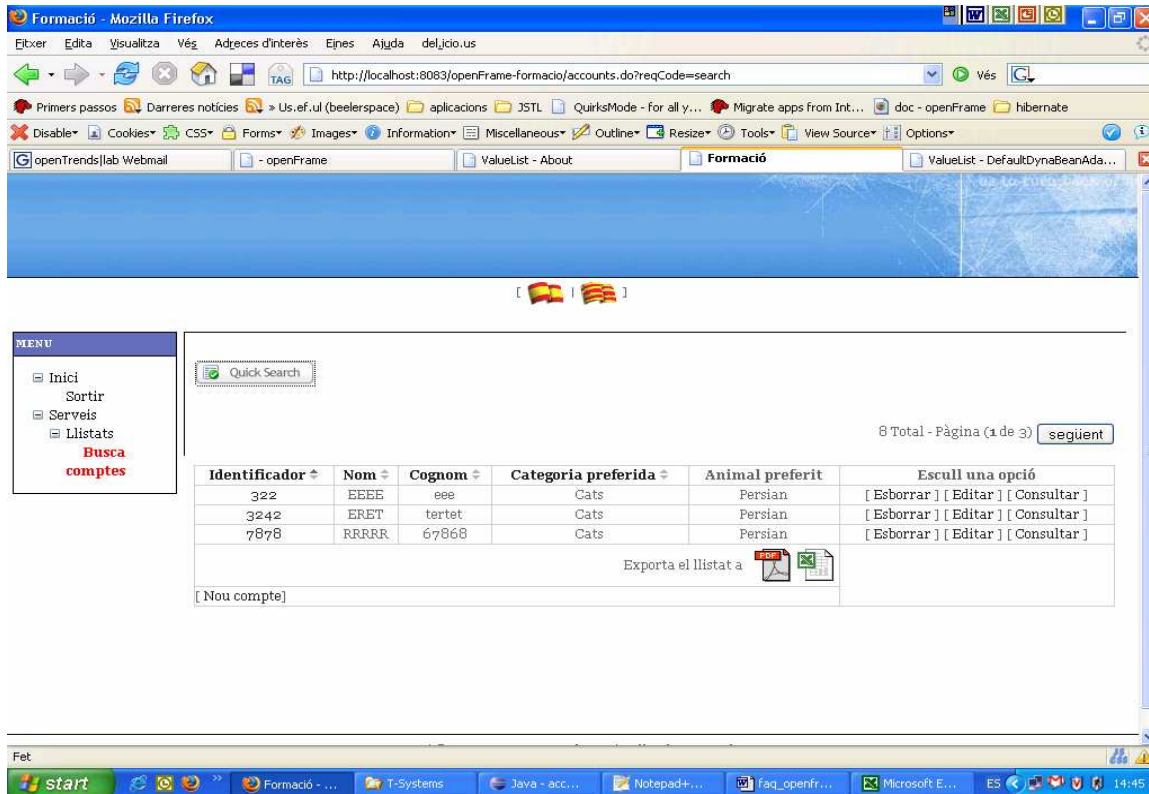
*posicio*: correspon, en aquest cas, a 0 o 1, depenent si volem recuperar informació de Account o de Product. L'ordre en el qual es troben dintre de l'array ve determinat per l'ordre en els quals els hem posat a la sentència SELECT.

*propietat*: propietat que volem recuperar.

Per exemple, en aquest cas tindriem:

```
<vlh:row bean="llista" display="<%=displayProvider%>">
...
<vlh:column titleKey="jsp.accounts.accountList.id" property="id"
sortable="desc">
<c:out value="{llista.array[0].id}"></c:out>
</vlh:column>
<vlh:column titleKey="jsp.accounts.accountList.firstname" property="firstname"
sortable="desc">
<c:out value="{llista.array[0].firstname}" />
</vlh:column>
<vlh:column titleKey="jsp.accounts.accountList.lastname" property="lastname"
sortable="asc">
<c:out value="{llista.array[0].lastname}" />
</vlh:column>
<vlh:column titleKey="forms.accountForm.field.preferredCategory"
property="preferredCategory.name" sortable="asc">
<c:out value="{llista.array[0].preferredCategory.name}" />
</vlh:column>
<vlh:column titleKey="forms.accountForm.field.preferredAnimal"
property="preferredProduct.name" sortable="asc">
<c:out value="{llista.array[1].name}" />
</vlh:column>
...
</vlh:row>
```

que ens genera el següent llistat:



Formació - Mozilla Firefox

http://localhost:8083/openFrame-formacio/accounts.do?reqCode=search

Primeros pasos Darrerres noticies Us.ef.ul (beelerspace) aplicacions JSTL QuirksMode - for all y... Migrate apps from Int... doc - openFrame hibernate

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

openTrends|lab Webmail - openFrame ValueList - About Formació ValueList - DefaultDynaBeanAda...

[ Espanya ] [ Espanya ]


MENU

- Inici
- Sortir
- Serveis
- Llistats
- Busca comptes

Quick Search

8 Total - Pàgina (1 de 3) [següent](#)

Identificador	Nom	Cognom	Categoria preferida	Animal preferit	Escull una opció
322	EEEE	eee	Cats	Persian	[ Esborrar ] [ Editar ] [ Consultar ]
3242	ERET	tertet	Cats	Persian	[ Esborrar ] [ Editar ] [ Consultar ]
7878	RRRR	67868	Cats	Persian	[ Esborrar ] [ Editar ] [ Consultar ]

Exporta el llistat a 

[ Nou compte ]

Fet

start Formació - ... T-Systems Java - acc... Notepad+... faq\_openfr... Microsoft E... ES 14:45



#### **1.4.7. L'exportació dels llistats a PDF i Excel només m'exporta la primera pàgina.**

##### **Problema**

El descrit a la pregunta

##### **Solució**

Aquest problema ha estat solucionat a partir de la versió 1.0.1-SNAPSHOT de la llibreria *openFrame-services-web.jar*

El comportament que es segueix ara és el següent:

- Si el paràmetre *maxExportRows* està informat i és diferent de zero, el nombre de registres que s'exporten és l'indicat per aquest paràmetre.
- Si el paràmetre no està informat o bé el seu valor és zero, s'exporta tots els registres del llistat.



#### 1.4.8. Com podem modificar els paràmetres de cerca que rep el servei de llistats de la Request?

##### Problema

L'objectiu és o bé afegir més paràmetres de filtre a la cerca dels llistats o bé modificar els paràmetres que venen a la request.

##### Solució

A partir de la versió 1.0.1-SNAPSHOT de la llibreria *openFrame-services-web.jar* està disponible una nova funcionalitat que permet modificar els paràmetres de cerca que rep el servei de generació de llistats, o bé afegir més paràmetres de cerca.

L'únic que cal fer és informar una Map amb els paràmetres que volem fer servir a la cerca, abans de fer la crida al mètode *search* del *valueListActionHelper* del nostre client.

Per exemple, suposem que volem que a totes les cerques que es facin es tingui en compte un paràmetre de nom *idioma* el qual no serà informat per pantalla sinó que serà informat segons certa casuística de negoci. Aleshores, el que farem serà el següent:

```
public ActionForward search(ActionMapping mapping, ActionForm form,
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response)
    throws Exception {

    /* Afegim un nou paràmetre de filtre a la cerca. */
    Map wParamsUser = new HashMap();
    String sIdioma = "CAT"; //valor per defecte

    // ..... casuística per a informar sIdioma

    wParamsUser.put("idioma",sIdioma);
    request.setAttribute(ValueListActionHelper.VALUE_LIST_FILTERS_ATTRIBUTE_NAME,
wParamsUser);

    valueListActionHelper.search(mapping,form,request,response);

    return mapping.findForward("success");
}
```

Fixem-nos que el hem de fer es ficar un attribute a la request de nom  
ValueListActionHelper.VALUE\_LIST\_FILTERS\_ATTRIBUTE\_NAME,  
i que guardi la Map amb els nous valors de cerca.

Si en aquesta Map existeix un paràmetre amb el mateix nom que un dels paràmetres de cerca de la request, aleshores en el moment de fer la cerca el valor que s'agafarà és el de la Map. És a dir, els valors de la Map tenen preferència respecte als de la Request. Això ens permet modificar els valors informats per l'usuari al filtre de cerca.



#### **1.4.9. Com podem afegir nous paràmetres de cerca per a les consultes que retornen les dades de les combos?**

##### **Problema**

Idem a l'anterior.

##### **Solució**

L'objectiu d'aquest punt és com podem informar nous paràmetres de cerca de les consultes associades a les combos (o Selects), els quals no venen informats des de el formulari. És a dir, són paràmetres de cerca que no venen informats a la request.

La idea és fer servir el mateix mecanisme descrit a la faq anterior, però en aquets cas el nomd e la Map és diferent. En lloc de ser

*ValueListActionHelper.VALUE\_LIST\_FILTERS\_ATTRIBUTE\_NAME*

hem de fer servir

*VlbOptionListSourceImpl.VALUE\_OPTION\_LIST\_FILTERS\_ATTRIBUTE\_NAME*

Per exemple, imaginem que tenim una combo en les quals les dades han de sortir filtrades per un paràmetre idioma que ha de ser informat des de l'aplicació i no per l'usuari.

La consulta que tenim associada a aquesta select és la següent:

```
<property name="hql">
  <value>
    FROM
    net.opentrends.openframe.formacio.model.Product
    AS vo WHERE 1=1
    /~category: AND vo.category.id LIKE {category} ~/
    /~idiomaAp: AND vo.idioma = {idiomaAp} ~/
    ORDER BY vo.id asc
  </value>
</property>
```

En aquest cas, el paràmetre category serà informat des de el formulari de dades i el paràmetre idiomaAp haurà de ser informat des de l'aplicació.

Per tal d'informar aquest paràmetre, des del mètode corresponent de l'action que s'encarregui de gestionar la petició, tindrem el següent codi:

```
..... Codi del mètode XXX de la classe Action ActionXXXX .....

/* Afegim un nou paràmetre de filtre a la cerca. */
Map wParamsUser = new HashMap();
String sIdioma = "CAT"; //valor per defecte
```



```
// ..... casuística per a informar sIdioma

wParamsUser.put("idiomaAp",sIdioma);

request.setAttribute(VlhOptionListSourceImpl.VALUE_OPTION_LIST_FILTERS_A
TTRIBUTE_NAME, wParamsUser);

....
```

Amb això el que estem fent és indicar que la llista de paràmetres de cerca de aplicar a la consulta ve donada per:

Els paràmetres de la Map informada a l'Action.  
Els paràmetres informats a la request.

És important tenir en compte el següent: si a la Map i a la request tenim dos paràmetres amb el mateix nom, alhora de generar la llista de paràmetres de cerca que s'aplicarà a la consulta, el valor del paràmetre a la request és el que s'aplicarà al final. És a dir, els valors dels paràmetres a la request predominen respecte als valors de la Map en cas de coincidència en els noms.





#### 1.4.10. Com puc implementar el meu propi custom editor?

##### Problema

El descrit a la pregunta

##### Solució

Anem a explicar amb un exemple com podem implementar un custom editor per a un tipus específic de dades.

En el nostre cas la dada que volem formatejar és un número de compte bancari, el qual per pantalla és informat per l'usuari en un camp de text i en canvi al nostre objecte de negoci la dada queda representada per un objecte de tipus *CompteBancari*, el qual té com a atributs: entitat bancaria, oficina, dígit de control i número de compte.

L'entitat *CompteBancari* ve representat per la següent pojo:

```
public class CompteBancari {  
  
    private String entitat;  
    private String oficina;  
    private String digits;  
    private String compte;  
  
    public CompteBancari() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
  
    public CompteBancari(String entitat, String oficina, String digits, String compte){  
        super();  
        // TODO Auto-generated constructor stub  
        this.entitat = entitat;  
        this.oficina = oficina;  
        this.digits = digits;  
        this.compte = compte;  
    }  
  
    // setters i getters....  
}
```

I al nostre objecte de negoci tindrem un atribut de tipus *CompteBancari*:

```
public class Account extends BaseAccount {  
    ....  
    private CompteBancari compteBancari;  
    ....  
}
```



El primer pas a fer es generar la classe custom editor. Aquesta classe ha d'extendre de *java.beans.PropertyEditorSupport*. I l'altra condició que ha de complir és redefinir el comportament del mètode *setAsText*.

En el nostre cas, tindrem la següent classe:

```
import java.beans.PropertyEditorSupport;

import net.opentrends.openframe.formacio.model.CompteBancari;

import org.springframework.util.StringUtils;

public class CompteBancariEditor extends PropertyEditorSupport {

    public CompteBancariEditor() {
        super();
        // TODO Auto-generated constructor stub
    }

    public CompteBancariEditor(Object source) {
        super(source);
        // TODO Auto-generated constructor stub
    }

    /**
     * Mètode que s'encarrega de transformar un String en un compte <br>
     * bancari. <br>
     * El format de la tribut que rep és: <br>
     * EEEEEOOODDDCCCCCCCCCCCC <br>
     * on: <br>
     * <li> EEEE: Representa a la entitat bancaria.
     * <li> OOOO: Representa a la oficina.
     * <li> DD: Representa els dígets del compte.
     * <li> CCCCCCCCCCCC: Representa el número de compte.
     */
    public void setAsText(String pText) throws IllegalArgumentException {
        if(!StringUtils.hasText(pText))
            throw new IllegalArgumentException("El paràmetre no pot ser null");

        String sEntitat = pText.substring(0,4);
        String sOficina = pText.substring(4,8);
        String sDigits = pText.substring(8,10);
        String sCompte = pText.substring(10);

        CompteBancari wCompte = new CompteBancari(sEntitat, sOficina, sDigits,
sCompte);

        setValue(wCompte);
    }
}
```



Fixem-nos que el que fem és formatjar la dada que ens ve com a paràmetre a un objecte de tipus `CompteBancari`. Al final fem la crida a `setValue(***)` que modifica el valor del paràmetre que estem editant (`compteBancari` en aquest cas).

Altra mètode interessant d'implementar és el `getAsText`, el qual ens permetrà indicar de quina manera s'ha de formatjar l'objecte quan sigui mostrat per pantalla. Per exemple.

Un cop tenim el custom editor definit, l'únic que cal fer és registrar-lo dintre de la llista de custom editors de l'aplicació. En el cas del framework, aquesta tasca es fa al fitxer `spring/property-editors.xml`.

Haurem d'afegir:

```
<bean id="customEditors" class="java.util.HashMap">
  <constructor-arg>
    <map>
      ...
      <entry key="net.opentrends.openframe.formacio.model.CompteBancari">
        <bean id="compteBancariEditor"
class="net.opentrends.openframe.formacio.util.propertyEditors.CompteBancariEditor"
"/>
      </entry>
    </map>
  </constructor-arg>
</bean>
```

on la *key* de la *entry* correspon al tipus d'objecte al qual volem associar un custom editor, i el valor de l'entrada correspon al custom editor a associar.



### 1.4.11. Com puc generar un desplegable amb les dades d'una taula mestre?

#### Problema

Volem generar un desplegable amb el framework i no sabem com fer-ho. Les dades a mostrar provenen d'una taula mestre.

#### Solució

Els passos a seguir per a generar un desplegable a partir de les dades d'una taula mestre són els següents:

- Definir la consulta HQL i configurar el bean del desplegable.
- Definir els atributs de visualització del desplegable.
- Definir el tag dintre de la JSP.

Per a descriure cada pas anem a definir un desplegable amb les dades de Categories.

#### Definir la consulta HQL i configurar el bean del desplegable

La definició d'aquesta consulta HQL l'hem de fer al fitxer  
*resources/spring/openFrame-services-web-options-lists.xml*.

La modificació que hem d'afegir al fitxer és la següent:

```
<bean name="defaultOptionValueListHandler"
      class="net.mlw.vlh.DefaultValueListHandlerImpl"> ❶
  <property name="config.adapters">
    <map>
      <entry key="categoriesList">
        <bean parent="defaultOptionBaseHibernateAdapter">
          <property name="hql">
            <value>
              FROM
                net.opentrends.openframe.formacio.model.Category
              AS vo WHERE l=1
              ORDER BY vo.id asc
            </value>
          </property>
        </bean>
      </entry>
    </map>
  </property>
</bean>

<bean id="categoriesOptionListSource" parent="defaultOptionListSource"> ❷
  <property name="optionListName" value="categoriesList"/>
  <property name="optionLabelName" value="name"/>
  <property name="optionLabelProperty" value="id"/>
</bean>

<bean
  id="optionsListService"
  class="net.opentrends.openframe.services.web.taglib.util.options.OptionsListServiceBase"> ❸
  <property name="optionsListSources">
    <map>
```



```
<entry key="categoriesList">
  <ref bean="categoriesOptionListSource"/>
</entry>
</map>
</property>
</bean>
```

Podem veure que hi ha definits 3 beans:

- ❶ En aquest pas, dintre de la propietat *config.adapters* del bean hem d'afegir l'entrada corresponent a la sentència HQL que correspon a com recuperar les dades del nostre desplegable.
- ❷ En aquest pas definim un bean que representa al nostre desplegable: indiquem quina consulta es farà servir (*optionListName*), quin serà l'identificador de cada element del desplegable (*optionLabelProperty*) i quina serà l'etiqueta que es mostrarà per a cada element (*optionLabelName*).
- ❸ Per últim, afegim el bean definit en el pas anterior a la llista de beans disponibles per a generar un desplegable. El nom amb el qual el registrem dintre d'aquest llistat es com l'hauem de referenciar després.

### **Definir els atributs de visualització del desplegable**

Aquest pas implica modificar el fitxer *spring/action-servlet-xxx.xml* associat al formulari on volem afegir el desplegable.

Dintre de la propietat *tags.configuration* del bean corresponent a la nostra acció, afegirem (a la entrada on ens interressi):

```
<bean parent="selectFieldTag">
  <property name="key" value="forms.accountForm.field.preferredCategory"/>
  <property name="styleId" value="preferredCategory.id"/>
  <property name="optionsListName" value="categoriesList"/> ❶
  <property name="layout" value="true"/>
  <property name="mode" value="E,E,E"/>
</bean>
```

Fixem-nos que el nom indicat a ❶ coincideix amb el definit en el pas ❸ anterior.

### **Afegir el tag a la JSP**

Aquest darrer pas simplement significa afegir el següent codi:

```
<fwk:select          otherKey="blank"          styleId="preferredCategory.id"
property="preferredCategory.id" />
```



#### 1.4.12. Error amb les connexions amb Tomcat: Socket closed

##### Problema

L'aplicació ens retorna el següent error:

*java.sql.SQLException: Excepción de E/S: Socket closed*

##### Solució

Depenent de la configuració que tinguem al nostre Datasource, pot ser que al demanar una connexió a aquest i executar qualsevol consulta ens retorni un missatge d'error on podem trobar la següent causa:

*java.sql.SQLException: Excepción de E/S: Socket closed*

Aquest error es produeix quan el datasource està retornant a l'aplicació una connexió que ja no és vàlida.

Per tal d'evitar aquest comportament del Datasource, els servidors d'aplicacions implementen un mecanisme alhora de definir un datasource que permet validar l'estat de la connexió abans de ser assignada a un client. En particular el que es fa és que abans de lliurar la connexió a un client, s'executa una consulta fent servir la connexió: si la consulta retorna resultats, aleshores la connexió és vàlida, i sinó retorna resultats o dona errors, la connexió és rebutja i s'agafa una altra per a lliurar-la al client.

En el cas particular del AS Tomcat, aquesta validació s'ha d'afegir a la configuració del DS que fem al fitxer *server.xml* (sempre hi quan tinguem la configuració del DS via el servidor). El codi a afegir és el següent:

```
<ResourceParams name="jdbc/formacioDS">
...
  <parameter>
    <name>validationQuery</name>
    <value>select count(*) from dual</value> ❶
  </parameter>
...
</ResourceParams>
```

❶ Aquest paràmetre correspon a la consulta que volem que s'executi per a validar la connexió. Es recomana fer servir aquesta o *SELECT 1 FROM DUAL*.



### 1.4.13. Exemple d'invocació client-servidor mitjançant AJAX

#### Problema

El descrit a la pregunta.

#### Solució

L'objectiu d'aquest document és donar un **exemple** d'ús d'AJAX. En particular, anem a veure com implementar un filtre per a les combos (o selects), de tal manera que l'usuari pugui filtrar el contingut d'aquestes simplement introduint paraules de filtrat.

Els passos a seguir són els següents:

- Modificar la consulta HQL que ens retorna el contingut de la combo, per tal d'afegir els criteris de filtre.
- Definir una classe Java que volem invocar desde el client.
- Registrar la classe dintre del fitxer de configuració de DWR.
- Iniciar el servidor i provar la implementació.
- Implementar la invocació a la nostra JSP.

#### Modificar consulta HQL

La consulta HQL de la nostra combo serà la següent (li hem afegit el criteri de cerca nom)

```
<entry key="categoriesList">
  <bean parent="defaultOptionBaseHibernateAdapter">
    <property name="hql">
      <value>
        FROM
        net.opentrends.openframe.formacio.model.Category
        AS vo WHERE 1=1
        /~nom: AND lower(vo.name) LIKE lower('[nom]%' ) ~/
        ORDER BY vo.id asc
      </value>
    </property>
  </bean>
</entry>
```

#### Definir classe Java

La única condició que ha de seguir aquesta classe és que implementi la interface *java.io.Serializable*.

En el nostre cas, volem que implementi un mètode que rebi el nom del llistat associat a la select i el paràmetre de cerca, i retorni un llistat amb els elements de la combo que verifiquin la condició de filtre.



Com la gestió de les combos al framework es delega al servei de llistats, aquesta classe té definit un atribut de tipus *OptionsListService*, que es correspondrà al bean que hi ha definit al fitxer de configuració *openFrame-services-web-list-options.xml*.

A continuació es mostra el codi de la classe:

```
package net.opentrends.openframe.formacio.util;

import java.io.Serializable;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import net.opentrends.openframe.services.web.taglib.util.options.OptionsListService;

public class OptionsUtil implements Serializable {

    private OptionsListService optionsListService;

    public OptionsUtil() {
        super();
        // TODO Auto-generated constructor stub
    }

    public List getOptionList(String pNameList, String pParamFiltre){
        Map wFiltre = new HashMap();
        wFiltre.put("nom",pParamFiltre);
        return optionsListService.getOptionsFromMap(pNameList, wFiltre);
    }

    public OptionsListService getOptionsListService() {
        return optionsListService;
    }
    public void setOptionsListService(OptionsListService optionsListService) {
        this.optionsListService = optionsListService;
    }
}
```

Un cop hem definit la classe, hem de registrar un bean dintre d'Spring que ens permeti injectar el valor de l'atribut.





Per fer això, definim el següent bean (una recomanació d'on registrar aquest bean pot ser al fitxer *account-servlet-xxx.xml* associat a l'acció):

```
<bean id="optionsUtilBean"
class="net.opentrends.openframe.formacio.util.OptionsUtil">
  <property name="optionsListService" ref="optionsListService"/>
</bean>
```

### **Definir la classe dintre del fitxer de configuració de DWR**

En aquest pas el que hem de fer és afegir dintre del fitxer de configuració de DWR la referència a la nostra classe (aquest fitxer indica quines classes pot crear i invocar DWR de manera remota des de javascript).

El codi a afegir-hi al fitxer */resources/dwr/dwr.xml* és:

```
<create creator="spring" javascript="optionsUtil" >
  <param name="beanName" value="optionsUtilBean"/>
</create>
```

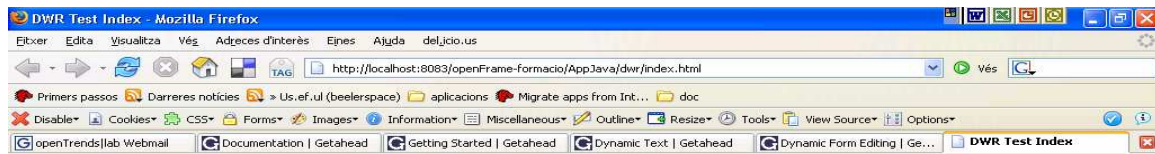
El paràmetre javascript correspon al nom mitjançant el qual farem referència al bean des de el nostre codi javascript.

### **Provar la implementació**

Per a provar que tots els passos anteriors els hem fet bé, simplement hem d'iniciar el servidor i anar al següent enllaç (al fitxer *web.xml* és on tenim definida que totes les peticions del tipus */AppJava/dwr/\** siguin gestionades pel servlet de DWR):

[http://\[host\]:\[port\]/\[YOUR-WEBAPP\]/AppJava/dwr/](http://[host]:[port]/[YOUR-WEBAPP]/AppJava/dwr/)

Haurem de veure una pantalla semblant a la següent:



### Classes known to DWR:

- [webOptionsListService](#) (net.opentrends.openframe.services.web.taglib.util.options.OptionsListServiceBase)
- [optionsUtil](#) (net.opentrends.openframe.formacio.util.OptionsUtil)
- [webValidationService](#) (net.opentrends.openframe.services.web.validation.common.CommonsWebValidationServiceImpl)

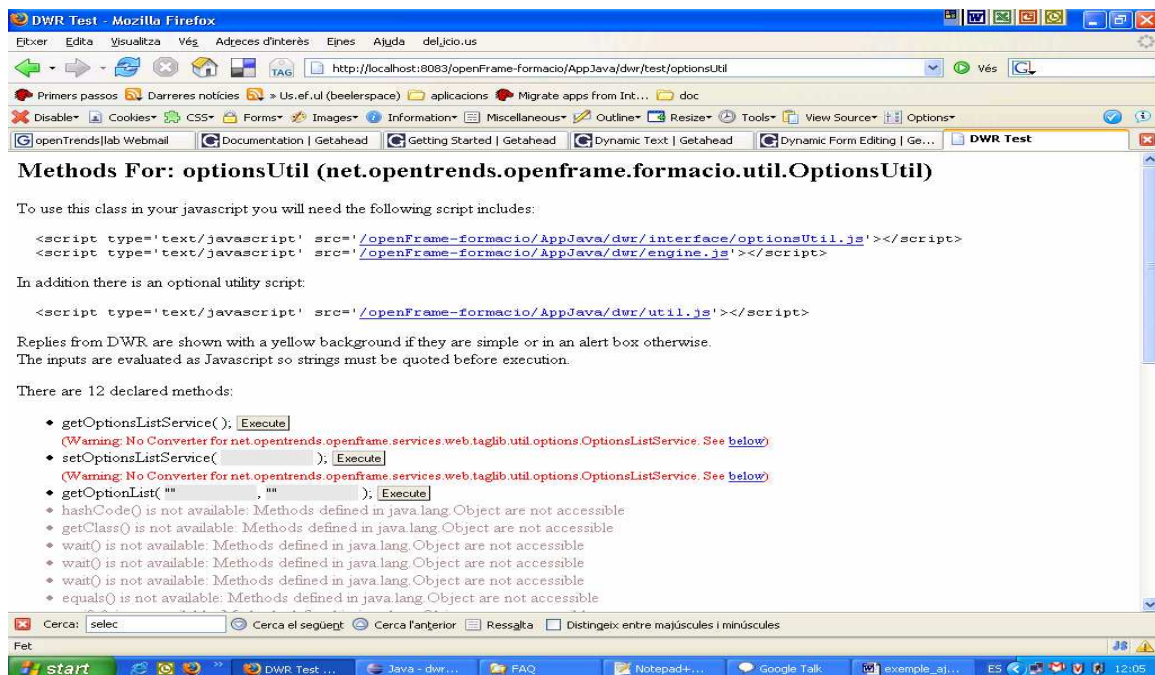
### Other Links

- Up to [top level of web app](#).



Si tot a anat bé, a la llista de classes conegudes per DWR ha de sortir la que hem definit prèviament.

Si ara clickem sobre l'enllaç de la nostra classe, veurem la següent pantalla:





En aquesta pantalla podem veure tots els mètodes de la classe que podem invocar des de javascript. I a més, podem provar la seva invocació des d'aquí.

Per exemple, anem a provar si la implementació que hem fet del mètode *getOptionList* és correcta. Per a això, informem els dos paràmetres d'entrada amb els valors "categoriesList" i "ca", i polsem el botó execute (ens ha de retornar tota la llista de categories que comencen per ca, aplicant ignoreCase).

Obtenim:

**Methods For: optionsUtil (net.opentrends.openframe.formacio.util.OptionsUtil)**

To use this class in your javascript you will need the following script includes:

```
<script type='text/javascript' src='/openFrame-formacio/AppJava/dwr/test/optionsUtil.js'></script>
```

In addition there is an optional utility script:

```
<script type='text/javascript' src='/openFrame-formacio/AppJava/dwr/test/optionsUtil.js'></script>
```

Replies from DWR are shown with a yellow background  
The inputs are evaluated as Javascript so strings must be

There are 12 declared methods:

- `getOptionsListService()`; [Execute]
- `setOptionsListService()`; [Execute]
- `getOptionList("categoriesList", "ca")`; [Execute]
- `hashCode()` is not available: Methods defined in java.lang.Object are not accessible
- `getClass()` is not available: Methods defined in java.lang.Object are not accessible
- `wait()` is not available: Methods defined in java.lang.Object are not accessible
- `wait()` is not available: Methods defined in java.lang.Object are not accessible
- `wait()` is not available: Methods defined in java.lang.Object are not accessible
- `equals()` is not available: Methods defined in java.lang.Object are not accessible

Modal dialog box content:

```
[
  {
    label:canario,
    value:CANARIO,
  },
  {
    label:Cats,
    value:CATS,
  },
]
```

[D'acord]

tal i com esperavem.

Un cop hem validat el funcionament, el que hem de fer és seguir les indicacions que ens dona DWR al començament de la pantalla: hem d'afegir 3 fitxers js a la nostra aplicació.

Dos d'ells ja els tenim: *engine.js* i *util.js* (s'afegeixen amb el tag del framework *fwk:configuration*).

I respecte al tercer fitxer, simplement hem de copiar la línia que apareix per pantalla

```
<script type='text/javascript' src='/openFrame-formacio/AppJava/dwr/interface/optionsUtil.js'></script>
```

i afegir-la a la nostra JSP (aquest js el genera DWR de manera dinàmica a l'inici del servidor).



### **Invocar implementació a la nostra JSP**

L'únic que queda ja per fer és implementar el codi javascript a la nostra JSP. En aquest cas el que volem es que quan es produeixi l'event *onKeyPress* sobre la nostra select, aquesta es recarregui mostrant només els elements que comencen per la cadena de caràcters introduïda per l'usuari.

Per tant, les modificacions a la JSP són 3:

#### *Afegir la llibreria JS*

```
<script type='text/javascript'
src='<%=request.getContextPath()%>/AppJava/dwr/interface/optionsUtil.js'>
</script>
```

#### *Afegir els events que volem controlar a la definició de la select*

```
<fwk:select otherKey="blank" styleId="preferredCategory.id"
property="preferredCategory.id" onkeypress="buscar_op(event)"
onblur="borrar_buffer()" onclick="borrar_buffer()" />
```

En aquest cas, capturem els events *onKeyPress* més els events *onblur* i *onclick*, els quals fan una crida a una funció js que inicialitza les variables js que controlen el procés.

#### *Afegir codi javascript*

El codi a afegir-hi és:

```
<script>
var digitos=10 //quantitat de digits cercats
var puntero=0
var buffer=new Array(digitos) //declaració de l'array Buffer
var cadena="" //cadena de caràcters introduïda per l'usuari.

function recarrega(valorFiltre){
    optionsUtil.getOptionList('categoriesList', valorFiltre,
ompleLlista);
}

function ompleLlista(data){
    DWRUtil.removeAllOptions("preferredCategory.id");
    DWRUtil.addOptions("preferredCategory.id", data,"value","label");
}

function buscar_op(event){

    // Lletre clickada per l'usuari
    var letra = String.fromCharCode(event.which)

    if(puntero >= digitos){
        cadena="";
        puntero=0;
    }
}
```



```
// si es presiona la tecla ENTER, s'inicialitza la select al
seu estat inicial

    if (event.keyCode == 13){
        borrar_buffer();
        recarrega(null);
    }
    else{
        buffer[puntero]=letra;
        cadena=cadena+buffer[puntero];
        puntero++;

        recarrega(cadena);
    }

    // invalida l'acció de clickat de tecla per a evitar cerca del
primer caràcter.
    event.returnValue = false;
}

function borrar_buffer(){
    cadena=" ";
    puntero=0;
}

</script>
```

La part important d'aquest codi són les funcions recarrega i ompleLlista.

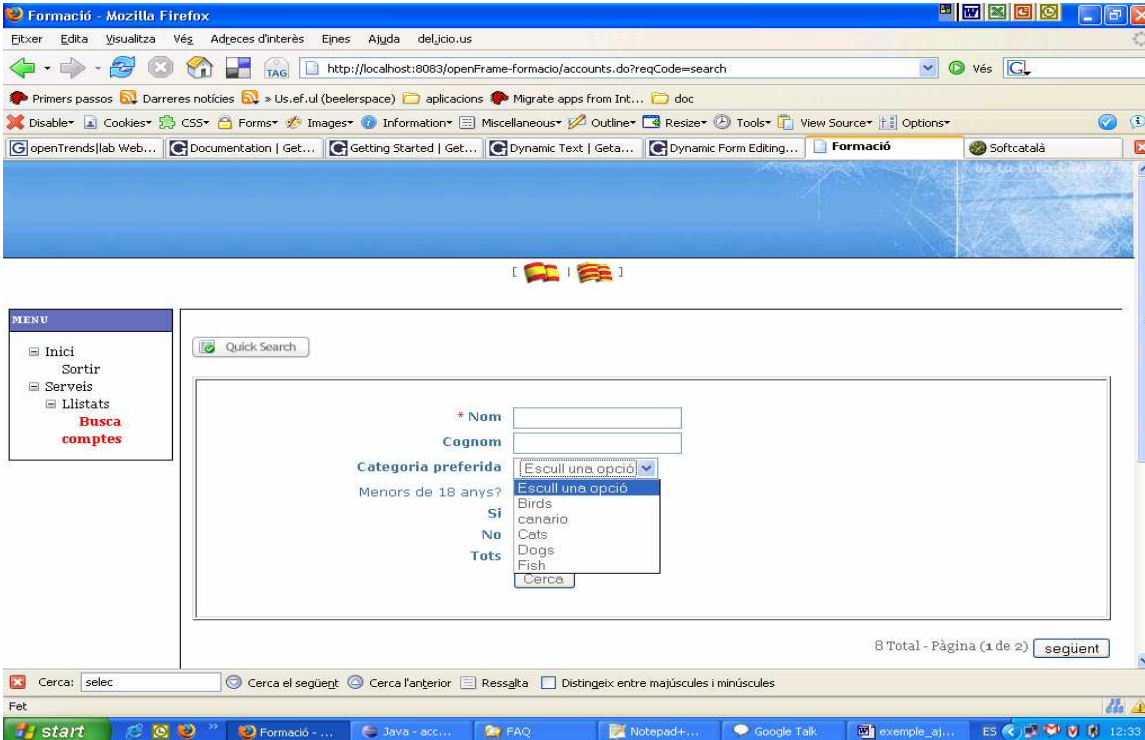
La primera és la que s'encarrega de fer la petició al servidor. Fixem-nos que fa una petició a un mètode que té 3 paràmetres: els dos primers corresponen als paràmetres que nosaltres hem definit a la nostra classe Java. I el tercer? El tercer correspon a la funció js a la qual ha de cridar quan rebí la resposta del servidor. Això es deu al fet a que la comunicació amb el servidor és asincrònica i per tant li hem d'indicar per on ha de continuar....

La segona funció és la que rep l'array de dades retornada pel servidor i s'encarrega d'omplir la select amb les noves dades. En aquest cas fa ús de crides a funcions js que ja venen implementades dintre de DWR (llibreria *util.js*).

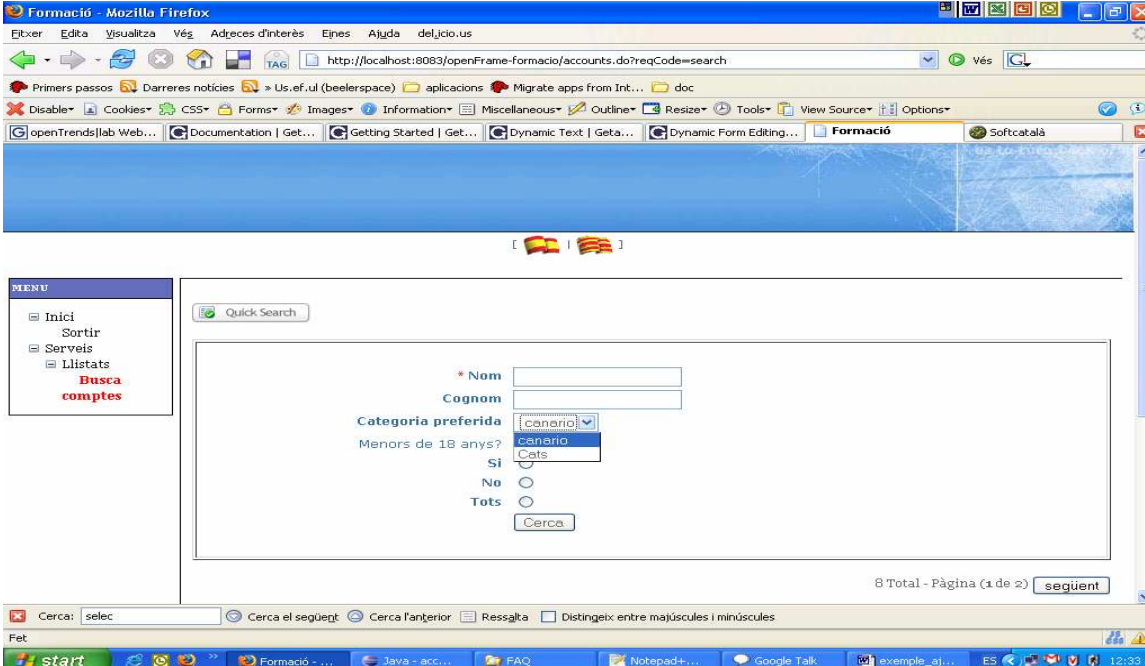
Amb tot això ja podem provar l'exemple: simplement ens hem de posicionar a la select i teclejar la paraula per la qual volem que filtri (les tecles polsades no es veuen per pantalla...).

A continuació podem veure el resultat d'executar l'exemple anterior.

Inicialment tenim la següent select:



Si ara ens situem sobre la select i polsem 'ca' el resultat és:







#### 1.4.14. Com modificar el comportament de la gestió d'errors que té DWR?

##### Problema

El descrit a la pregunta.

##### Solució

Per defecte, tot error que es produeix en la comunicació client-servidor mitjançant l'ús de la interfície DWR es mostra a l'usuari mitjançant un alert de javascript.

Això es deu a que per defecte DWR té definida la següent instrucció al fitxer engine.js

```
/**
 * A function to call if something fails.
 * @private
 */
DWREngine._errorHandler = DWREngine.defaultMessageHandler;
```

És a dir, està indicant que qualsevol error ha de ser gestionat per la funció js *defaultMessageHandler*, el codi de la qual és:

```
/**
 * The default message handler.
 * Useful in calls to setErrorHandler() or setWarningHandler() to allow
 you to
 * get the default back.
 * @param message The message to display to the user somehow
 */
DWREngine.defaultMessageHandler = function(message) {
    if (typeof message == "object" && message.name == "Error" &&
message.description) {
        alert("Error: " + message.description);
    }
    else {
        alert(message);
    }
};
```

Si a la nostra aplicació volem definir un gestor diferent pels errors, l'únic que hem de fer és indicar quina funció ha de ser el nou gestor. I això ho farem amb la següent crida al nostre codi:

```
<script>DWREngine.setErrorHandler=nom_de_la_funcio; </script>
```

On *nom\_de\_la\_funcio* és una funció javascript que tindrem definida, per exemple:

```
function nom_de_la_funcio(message){
    alert("S'ha produït el següent error:"+message);
}
```



### 1.4.15. Com puc afegir un CustomEditor per les dades de tipus Number?

#### Problema

Es vol definir un Custom Editor per fer el mapeig de un string a un objecte de tipus Number, per exemple Integer, en el procés de binding de les dades.

#### Solució

Spring ens proporciona una classe que ja fa aquest procés, CustomNumberEditor. Per a registrar-la dintre dels customs editors de la nostra aplicació el que s'ha de fer és afegir-la dintre del fitxer property-editor.xml.

En concret, hem d'afegir el següent codi:

```
<entry key="java.lang.Integer">
  <bean id="integerEditor"
class="org.springframework.beans.propertyeditors.CustomNumberEditor" >
    <constructor-arg type="java.lang.Class">
      <value>java.lang.Integer</value> ❶
    </constructor-arg>
    <constructor-arg>
      <value>false</value> ❷
    </constructor-arg>
  </bean>
</entry>
```

On

- ❶ Correspon al tipus de dades que volem tractar. Ha de ser una classe de tipus Number.
- ❷ Si volem permetre valors nulls o buits. Si tenim activat aquest flag el que fa el custom editor és assignar un null a l'atribut quan des de el formulari ve el camp buit. Si volem que assigni un valor per defecte aleshores es recomana estendre el customEditor i redefinir el mètode setAsText.





### 1.4.16. Problemes amb els elements enllaçats o dependents

#### Problema

L'enllaçament d'elements no funciona.

#### Solució

A la nostra plana tenim elements enllaçats de tal manera que quan es modifica el valor d'un és recarrega el valor de l'altre.

Per exemple, tenim dos selects *selectA* i *selectB*, i quan modifiquem el valor de la *selectA* es recarrega el valor de la *selectB*, és a dir el valor seleccionat a la *selectA* ens permet filtrar els elements de la *selectB*.

Si hem implementat aquesta funcionalitat i veiem que no ens funciona, podem validar els següents aspectes de la implementació.

#### Definició de la dependència correcta

Al fitxer *action-servlet-xxx.xml*, on tinguem definida la dependència, hem de tenir definida la relació de la següent manera:

```
<bean parent="selectFieldTag"> // Configuració del selectA
  <property name="key" value="forms.accountForm.field.preferredCategory"/>
  <property name="styleId" value="preferredCategory.id"/>
  <property name="optionsListName" value="categoriesList"/>
  <property name="layout" value="true"/>
  <property name="mode" value="E,E,I"/>
</bean>
<bean parent="selectFieldTag"> // Configuració del selectB
  <property name="key" value="forms.accountForm.field.preferredAnimal"/>
  <property name="styleId" value="preferredProduct.id"/>
  <property name="optionsListName" value="animalsList"/>
  <property name="layout" value="true"/>
  <property name="mode" value="E,E,I"/>
  <property name="selectFieldSource"> ❶
    <map>
      <entry>
        <key><value>source</value></key>
        <value>preferredCategory.id</value>
      </entry>
      <entry>
        <key><value>paramName</value></key>
        <value>category</value>
      </entry>
    </map>
  </property>
</bean>
```

- ❶ Aquesta propietat ens diu que aquesta select té una dependència d'un altre element, l'identificador del qual ve donat pel valor de l'entrada *source* (en aquest cas correspon a una altra select).  
L'entrada *paramName* ens indica amb quin nom es passarà al filtre de la *selectB* el valor escollit a la *selectA*.

#### Definició de la consulta que ens retorna els valors de la selectB



En aquest cas haurem de validar que la consulta que ens retorna els valors de la *selectB* està ben definida i té com paràmetre de filtre l'element que ens ve de la *selectA*.

Per exemple, en el nostre exemple aquesta consulta correspon a:

```
FROM
net.opentrends.openframe.formacio.model.Product
AS vo WHERE l=1
/~category: AND vo.category.id LIKE {category} ~/ ❶
ORDER BY vo.id asc
```

- ❶ Podem observar que aquesta consulta té un paràmetre de filtre de nom *category*, el qual ha de coincidir amb el nom definit a la configuració anterior.

#### *Implementació correcta a la JSP*

L'ordre en el qual han d'apareixer els elements a la nostra plana JSP ha de ser el mateix ordre jeràrquic de la dependència, **sinó la dependència no funcionarà**.

És a dir, al nostre exemple, a la JSP els elements han d'estar implementats en el següent ordre:

```
...
<fwk:select otherKey="blank" styleId="preferredCategory.id"
property="preferredCategory.id"/>
<fwk:select styleId="preferredProduct.id" property="preferredProduct.id"/>
...
```

La raó per la qual deixa de funcionar la podem resumir en el següent: en el moment en el qual es renderitza el component JSP depenent, es valida que existeixi l'element html del qual depèn, i sinó existeix no es modifica el comportament de l'event *onchange*, i per tant els valors de la *selectB* no es recarreguen al modificar la *selectA*.

Podem mirar el codi següent al js *openFrame-ajaxtags-select.js*

```
if (this.options.sourceElem) {
    this.attachBehaviors(this.options.sourceElem, this.options.eventType,
        this.actionElemChanged, this);
    this.actionElemChanged(new Object());
}
```



### 1.4.17. Més d'un formulari en una mateixa JSP.

#### Problema

Definir més d'un formulari dins d'una mateixa JSP.

#### Solució

Haurem de seguir els següents passos:

1-Primerament haurem d'actualitzar dos fitxers a la release 1.0.8 o superior:

- openFrame-services-web-1.0.8.jar
- openFrame-ajaxtags-validation.js

2-Definirem les validacions dels formularis en el fitxer validation.xml per separat. En l'exemple que detallem definim 2 formularis:

```
<form-validation>

  <formset>

    <form name="accounts1">
      <field property="firstname" depends="required">
        <arg0 key="forms.accountForm.field.firstname"/>
        <arg1 key="forms.accountForm.field.lastname"/>
      </field>

      <field property="preferredCategory.id" depends="required">
        <arg0 key="forms.accountForm.field.preferredCategory"/>
      </field>
      <field property="email" depends="email">
        <arg0 key="forms.accountForm.field.email"/>
      </field>
      <field property="lastname" depends="required,maxlength">
        <arg0 key="forms.accountForm.field.lastname"/>
        <arg1 name="maxlength" key="{var:maxlength}"
              resource="false" />
        <var>
          <var-name>maxlength</var-name>
          <var-value>10</var-value>
        </var>
      </field>
      <field property="password" depends="required">
        <arg0 key="forms.accountForm.field.password"/>
      </field>
      <field property="addr1" depends="required">
        <arg0 key="forms.accountForm.field.addr1"/>
      </field>
      <field property="city" depends="required">
        <arg0 key="forms.accountForm.field.city"/>
      </field>
      <field property="state" depends="required,minlength">
        <arg0 key="forms.accountForm.field.state"/>
        <arg1 name="minlength" key="{var:minlength}"
              resource="false" />
        <var>
          <var-name>minlength</var-name>
          <var-value>1</var-value>
        </var>
      </field>
    </form>
  </formset>
</form-validation>
```



```
<field property="zip" depends="required">
  <arg0 key="forms.accountForm.field.zip"/>
</field>
<field property="country" depends="required">
  <arg0 key="forms.accountForm.field.country"/>
</field>
<field property="password" depends="required">
  <arg0 key="forms.accountForm.field.password"/>
</field>
<field property="comments" depends="required">
  <arg0 key="forms.accountForm.field.password"/>
</field>

</form>

<form name="accounts2">
  <field property="phone" depends="required">
    <arg0 key="forms.accountForm.field.phone"/>
  </field>
</form>

</formset>
</form-validation>
```

3-Dins del fitxer action-servlet-xxx.xml definirem un formTag per cada formulari:

```
...
<property name="tagsConfiguration">
  <map>
    <entry key="*">
      <list>

        <bean parent="formTag">
          <property name="styleId" value="actionForm"/>
          <property name="validationProperties">
            <props>
              <prop key="validationType">SERVER</prop>
              <prop key="validatorName">accounts1</prop>
              <prop key="pojoClass">net.openframe.Account</prop>
            </props>
          </property>
        </bean>
        <bean parent="formTag">
          <property name="styleId" value="actionForm2"/>
          <property name="validationProperties">
            <props>
              <prop key="validationType">SERVER</prop>
              <prop key="validatorName">accounts2</prop>
              <prop key="pojoClass">net.openframe.Account</prop>
            </props>
          </property>
        </bean>
      </list>
    </entry>
  </map>
</property>
...
```



Dins de validationProperties definirem les següents propietats:

- *validationType*: tipus de validació .Els possibles valors són CLIENT/SERVER.
- *validatorName*: nom del validador que hem donat dins del fitxer validation.xml.
- *pojoClass*: classe del Bean del Formulari.

4- En la JSP definirem els formularis amb el seu corresponent styleId (corresponent al definit dins del seu formTag de action-servlet-xxx.xml):

```
<fwk:form action="editAccount.do" styleId="actionForm" reqCode="edit" width="500"
method="post" styleClass="edit">
  <tr>
    <fwk:text styleId="id" property="id"/>
  </tr>
  <tr>
    <fwk:text styleId="password" property="password"/>
  </tr>
  <tr>
    <fwk:text styleId="email" property="email"/>
  </tr>
  <tr>
    <fwk:text styleId="firstname" property="firstname"/>
  </tr>
  ...
  <tr>
    <fwk:submit styleId="saveActionImage"/>
  </tr>
</fwk:form>

<fwk:form action="editAccount.do" styleId="actionForm2" reqCode="edit" width="500"
method="post" styleClass="edit">
  <tr>
    <fwk:text styleId="phone" property="phone"/>
  </tr>
  <tr>
    <fwk:submit styleId="saveActionImage2"/>
  </tr>
</fwk:form>

...
```



### 1.4.18. Eliminar columnes d'un llistat a l'exportar a excel o pdf.

#### Problema

Eliminar columnes d'un llistat a l'exportar a excel o pdf. Poder eliminar columnes per títol de la columna o per index de la columna.

#### Solució

Eliminar columnes per títol de columna:

Dins del fitxer openFrame-services-web-lists.xml hi trobem el bean de configuració del valuelist ([ValueListConfigBean](#)). Aquest bean conté els displayProviders ExportPdf i ExportExcel. Aquests beans tenen una propietat que es diu skipColumns. Aquí haurem d'introduir el titleKey de la columna que volem ignorar a l'exportar (tant a pdf com a excel):

```
<entry key="ExportPDF">
  <bean
class="net.opentrends.openframe.services.web.vlh.tag.support.PdfDisplayProvider"
singleton="false">
  <property name="logService" ref="loggingService"/>
  <property name="il8nService" ref="il8nService"/>
  <property name="skipColumns">
    <list>
      <value>jsp.categories.categoryList.actions</value>
      <value>jsp.products.productList.actions</value>
    </list>
  </property>
</bean>
</entry>
<entry key="ExportExcel">
  <bean
class="net.opentrends.openframe.services.web.vlh.tag.support.ExcelDisplayProvider"
singleton="false">
  <property name="logService" ref="loggingService"/>
  <property name="il8nService" ref="il8nService"/>
  <property name="skipColumns">
    <list>
      <value>jsp.categories.categoryList.actions</value>
      <value>jsp.products.productList.actions</value>
    </list>
  </property>
</bean>
</entry>
```

Eliminar columnes per index de columna:

Per eliminar columnes pel seu index, haurem de configurar-ho en la Action que fa l'export:

```
public ActionForward searchExportPDF(ActionMapping mapping, ActionForm form,
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response) throws Exception {
```



```
List skipColumnNumbers = new ArrayList();
skipColumnNumbers.add("1");
skipColumnNumbers.add("3");

request.setAttribute(PdfDisplayProvider.SKIP_COLUMN_NUMBERS, skipColumnNumbers);

request.setAttribute("displayProvider", "ExportPDF");

return this.search(mapping, form, request, response);
}

public ActionForward searchExportExcel(ActionMapping mapping, ActionForm form,
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response) throws Exception {

    List skipColumnNumbers = new ArrayList();
    skipColumnNumbers.add("1");
    skipColumnNumbers.add("3");

    request.setAttribute(ExcelDisplayProvider.SKIP_COLUMN_NUMBERS, skipColumnNumbers
);

    request.setAttribute("displayProvider", "ExportExcel");

    return this.search(mapping, form, request, response);
}
```

En l'exemple anterior, a l'exportar tant a pdf com a excel, estariem ignorant les columnes primera i tercera.



#### 1.4.19. Més d'una ValueList en una mateixa JSP.

##### Problema

Configurar una aplicació per poder mostrar més d'una ValueList en una mateixa JSP.

##### Solució

Els passos a seguir són els següents:

·Dins del fitxer action-servlet-xxx.xml, definir un Map d'objectes ValueListActionHelper, amb una clau que servirà per identificar cada ValueListActionHelper:

```
<bean name="/accounts" parent="accountBaseDefinition">
  <property name="valueListActionHelperMap">
    <map>
      <entry key="accounts">
        <bean parent="valueListActionHelper">
          <property name="listName">
            <value>accountList</value>
          </property>
          ❶<property name="listAttributeName"
            value="list" />
          ❷<property name="id" value="valuelist1"/>
          <property name="tableId"
            value="ACCOUNT" />
        </bean>
      </entry>
      <entry key="categories">
        <bean parent="valueListActionHelper">
          <property name="listName">
            <value>categoriesList</value>
          </property>
          <property name="listAttributeName"
            value="list2" />
          <property name="id" value="valuelist2"/>
          <property name="tableId"
            value="CATEGORY" />
        </bean>
      </entry>
    </map>
  </property>
  ...
</bean>
```

Per a cada ValueListActionHelper haurem de definir, a part de les que ja informavem (*listName* i *tableId*), les propietats ❶ *listAttributeName* (nom de la valuelist) i ❷ *id* (id de la valuelist).

·Dins l'Action definirem el Map d'objectes ValueListActionHelper i recuperarem els que necessitem per fer les búsquedes:

```
private Map valueListActionHelperMap;

...
```





```
public ActionForward search(ActionMapping mapping, ActionForm form,
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response)
    throws Exception {

    ValueListActionHelper valueListActionHelperAccounts =
    (ValueListActionHelper)valueListActionHelperMap.get("accounts");
    valueListActionHelperAccounts.search(mapping, form, request, response);

    ValueListActionHelper valueListActionHelperCategories =
    (ValueListActionHelper)valueListActionHelperMap.get("categories");
    valueListActionHelperCategories.search(mapping, form, request, response);

    FormUtils.setFormDisplayMode(request, form, FormUtils.EDIT_MODE);

    return mapping.findForward("success");
}
```

·En la JSP definirem les diferents valuelists posant com a *name* el nom que haguem definit en la propietat *listAttributeName* i *id* que també haguem definit en la propietat *id* dins l'action-servlet-xxx.xml:

```
<fwk:vlhroot ❶value="list" ❷id="valuelist1" url="accounts.do?"
includeParameters="reqCode" configName="vlConfig">
...
</fwk:vlhroot>

<vlh:root value="list2" id="valuelist2" url="categories.do?"
includeParameters="reqCode" configName="vlConfig">
...
</fwk:vlhroot>
```

·Exemple de JSP complet:

```
<%@ include file="/WEB-INF/jsp/includes/fwkJtagLibs.jsp" %>

<fwk:vlhroot value="list" id="valuelist1" url="accounts.do?"
includeParameters="reqCode" configName="vlConfig">
    <fwk:form styleId="actionForm" action="accounts.do" reqCode="search">

        <table width="450" align="center" border="0">
            <!-- pagination section -->
            <tr>
                <td align="left" nowrap="true"><c:out
                    value="${list.valueListInfo.totalNumberOfEntries}" />
                <fmt:message key="jsp.accounts.accountList.total"/> - <fmt:message
                    key="jsp.accounts.accountList.page"/>
                (<c:out value="${list.valueListInfo.pagingPage}" /> <fmt:message
                    key="jsp.accounts.accountList.of"/> <c:out
                    value="${list.valueListInfo.totalNumberOfPages}" />)</td>
```



```
<td><table border="0" align="right"><tr><td><fwk:vlhpaging
/></td></table></td>
</tr>
<!-- end of Pagination -->
<tr>
    <td colspan="2">
        <table width="100%" class="classicLook" cellspacing="0"
            cellpadding="0" id="taulaId">
            <tbody>
                <%
                    String displayProvider =
(request.getAttribute("displayProvider")!=null)?request.getAttribute("displayProvi
der").toString():"";
                <%>
                    <fwk:vlhrow bean="account"
display="<%=displayProvider%>">

                        <vlh:attribute
name="onmouseover">this.className='selected';</vlh:attribute>
                        <vlh:attribute
name="onmouseout">this.className='zebra0';</vlh:attribute>
                        <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.id" property="id">
                            <fwk:vlhaction url="editAccount.do?">
                                <fwk:vlhaddParam name="reqCode">

value="edit" temp="false" />
                                <fwk:vlhaddParam name="id"

property="id" temp="false" />
                                <c:out

value="${account.id}"></c:out>
                            </fwk:vlhaction>
                        </fwk:vlhcolumn>
                        <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.firstname" property="firstname"/>
                        <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.lastname" property="lastname"/>
                        <fwk:vlhcolumn
titleKey="forms.accountForm.field.preferredCategory"
property="preferredCategory.id"/>

                    </fwk:vlhrow>
                <tr>
                    <td colspan="4" style="text-align:right">
                        <table align="right"><tr><td
style="border: 0px"><fmt:message key="jsp.common.export"/>&nbsp;</td>
                        <td style="border: 0px"><html:link
page="/accounts.do?reqCode=searchExportPDF" target="_blank">" border="0"/></html:link></td>
                        <td style="border: 0px"><html:link
page="/accounts.do?reqCode=searchExportExcel" target="_blank">" border="0"/></html:link></td>
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
</table>

</fwk:form>

</fwk:vlhroot>
<br>
<vlh:root value="list2" id="valuelist2" url="categories.do?"
includeParameters="reqCode" configName="vlConfig">
    <table width="450" align="center">
```



```
<!-- pagination section -->
<tr>
    <td align="left" nowrap="true"><c:out
        value="${list2.valueListInfo.totalNumberOfEntries}" />
<fmt:message key="jsp.categories.categoryList.total"/> - <fmt:message
key="jsp.categories.categoryList.page"/>
        (<c:out value="${list2.valueListInfo.pagingPage}" />
<fmt:message key="jsp.categories.categoryList.of"/> <c:out
        value="${list2.valueListInfo.totalNumberOfPages}"
/>)</td>
    <td align="right"><vlh:paging/></td>
</tr>
<!-- end of Pagination -->
<tr>
    <td colspan="2">
<table width="100%" class="classicLook" cellspacing="0"
    cellpadding="0">
    <%
        String displayProvider2 =
(request.getAttribute("displayProvider") != null)?request.getAttribute("displayProvi
der").toString(): "ExportHTML";
    <%
        <vlh:row bean="category" display="<%=displayProvider2%>">
            <vlh:attribute
name="onmouseover">this.className='selected';</vlh:attribute>
            <vlh:attribute
name="onmouseout">this.className='zebra0';</vlh:attribute>
            <vlh:column
titleKey="jsp.categories.categoryList.id" property="id">
                <vlh:action url="categories.do?">
                    <vlh:addParam name="reqCode"
value="edit" temp="false" />
                    <vlh:addParam name="id"
property="id" temp="false" />
                <c:out
value="${category.id}"></c:out>
            </vlh:action>
            </vlh:column>
            <vlh:column
titleKey="jsp.categories.categoryList.desc" property="descn" sortable="desc" />
            <vlh:controls
titleKey="jsp.categories.categoryList.actions">
                [
                    <vlh:action url="products.do?">
                        <vlh:addParam name="reqCode"
value="search" temp="false" /> <fmt:message
key="jsp.categories.categoryList.see_products"/>
                        <vlh:addParam name="category" property="id"
temp="false" />
                        <vlh:addParam name="name" value="" temp="false"
/>
                    </vlh:action>
                ]
            </vlh:controls>
        </vlh:row>
    </td>
    <td colspan="3" style="text-align:right">
<table align="right"><tr><td
style="border: 0px"><fmt:message key="jsp.common.export"/>&nbsp;</td>
        <td style="border: 0px"><html:link
page="/categories.do?reqCode=searchExportPDF" target="_blank">" border="0"/></html:link></td>
        <td style="border: 0px"><html:link
page="/categories.do?reqCode=searchExportExcel" target="_blank">" border="0"/></html:link></td>
    </tr>
</td>
</tr>
```



```
                </table>  
            </td>  
        </tr>  
    </table>  
</v1h:root>
```