Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

# Framework Corporatiu J2EE

# Servei de configuració

**Versió 1.4**

Barcelona, 13 / abril / 2007

# Històric de modificacions

| Data | Autor | Comentaris | Versió |
|---|---|---|---|
| 09/12/2005 | Atos Origin, sae openTrends | Versió inicial del document | 1.0 |
| 12/01/2006 | Atos Origin, SAE | Versió 1.1 d'OpenFrame | 1.1 |
| 13/02/2007 | Atos Origin, SAE | Versió 1.2 d'OpenFrame | 1.2 |
| 16/03/2007 | Atos Origin, SAE | Versió 1.3 d'OpenFrame | 1.3 |
| 13/04/2007 | Atos Origin, SAE | Versió 1.4 d'OpenFrame | 1.4 |
| | | | |
| | | | |
| | | | |
| | | | |

**Llegenda de Marcadors**

# Índex

# 1. Introducció

## 1.1. Propósit

El Servei de Configuració té com a propósit la configuració de les propietats de qualsevol component de l'aplicació. Aquestes propietats poden ser tant referències a altres objectes com propietats internes (atributs) que necessiten per al seu correcte funcionament.

Aquesta configuració es pot realitzar de 2 formes diferenciades:

1) Forma tradicional

   Una de les formes tradicionals d'estructurar el nostre codi és que siguin els components qui realitzin la localització i instanciació dels serveis o components que han de cridar dins la seva lógica (així com l'obtenció de les propietats internes)

   En aquesta aproximació, encara utilitzada però poc pràctica, es pot fer ús del patró factory per amagar la complexitat de la instanciació i obtenció de les instàncies requerides. Tot i això, aquest mecanisme l'han de realitzar els clients, els quals han de conéixer quines dependències tenen els components entre sí i per tant conéixer quina seqüència han de seguir per la inicialització de cadascun d'ells.

   En altres casos, l'especificació J2EE fa ús de JNDI com a mecanisme d'obtenció de referències d'objectes. Aquesta implementació requereix molts canvis si es fa un canvi d'entorn i deixa d'usar-se JNDI.

2) Patró 'Dependency Injection'

   Mitjançant l'ús del patró Dependency Injection el nostre aplicatiu només ha de definir les dependències, i deixar que un codi extern (el framework) s'encarregui de la complexitat d'instanciació, inicialització i seqüència de les referència que requereixin els nostres components. Aquest patró elimina totes les problemàtiques de la forma tradicional.

openFrame es basa en l'ús del patró 'Dependency Injection' mitjançant l'ús de Spring. El més important és que el seu ús no és intrusiu, ja que en qualsevol moment podríem canviar de mecanisme intern sense afectar el nostre codi.

## 1.2. Context i Escenaris d'Ús

El Servei de Configuració es troba dins dels serveis de Propósit General de openFrame.

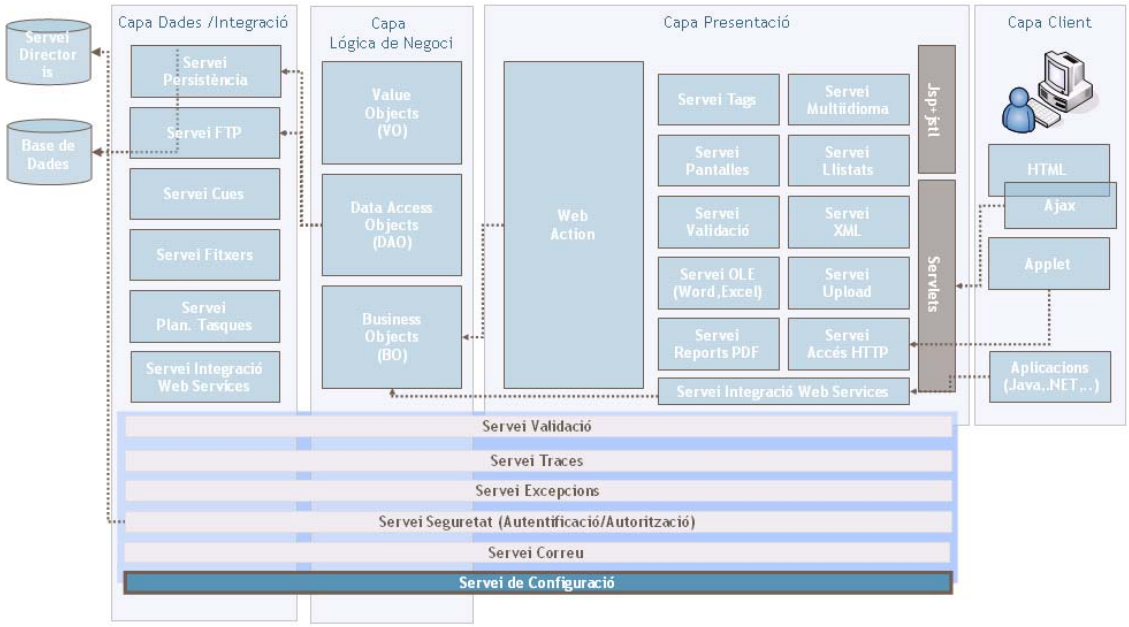
El seu ús és imprescindible en openFrame, ja que tots els serveis utilitzen la gestió de configuració definida per aquest servei.

## 1.3. Versions i Dependències

En el present apartat es mostren quines són les versions i dependències necessàries per fer ús del Servei.

Dins la llista de dependències es mostren diferenciades:

- Dependències bàsiques. Llibreries necessàries per fer ús del servei
- Dependències adicionals. Aquestes dependències són necessàries per poder fer ús de característiques concretes del servei o per l'ús dels tests unitaris proporcionats amb el servei




### 1.3.1. Versions

No s'han produït canvis respecte la versió 1.3.

### 1.3.2. Dependències Bàsiques

| Nom | Tipus | Versió | Descripció |
|---|---|---|---|
| openFrame-services-exceptions | jar | 1.0 | |
| openFrame-services-logging | jar | 1.0 | |
| spring | jar | 1.2.5 | http://www.springframework.org |

Per cada servei openFrame fer ús també de les dependències requerides per ell.

### *1.3.3. Dependències Adicionals*

- Proves Unitàries del Servei

| Nom | Tipus | Versió | Descripció |
|---|---|---|---|
| junit | jar | 3.8.1 | |

## 1.4. A qui va dirigit

Es recomana una lectura prèvia de 'http://static.springframework.org/spring/docs/1.2.x/reference/beans.html'

## 1.5. Documents i Fonts de Referència

Per a començar a fer ús del Servei de Configuració és imprescindible la lectura dels següents documents:

| Nom | Descripció |
|---|---|
| Configuració amb Spring | http://static.springframework.org/spring/docs/1.2.x/reference/beans.html |

I es recomana la lectura de les següents referències:

| Nom | Descripció |
|---|---|
| Introducció patró 'Dependency Injection' | http://www.theserverside.com/articles/article.tss?l=IOCBeginners |
| Definició del patró 'Dependency Injection' | http://www.martinfowler.com/articles/injection.html |

## 1.6. Glossari

**Dependency Injection**

Patró de disseny que estableix un nivell d'abstracció mitjançant la definició de interfícies i eliminant la dependència entre components mitjançant un codi extern que injecta aquestes dependències de forma transparent. Existeixen 3 formes del patró: 'setter', 'constructor' i 'interface based injection'.

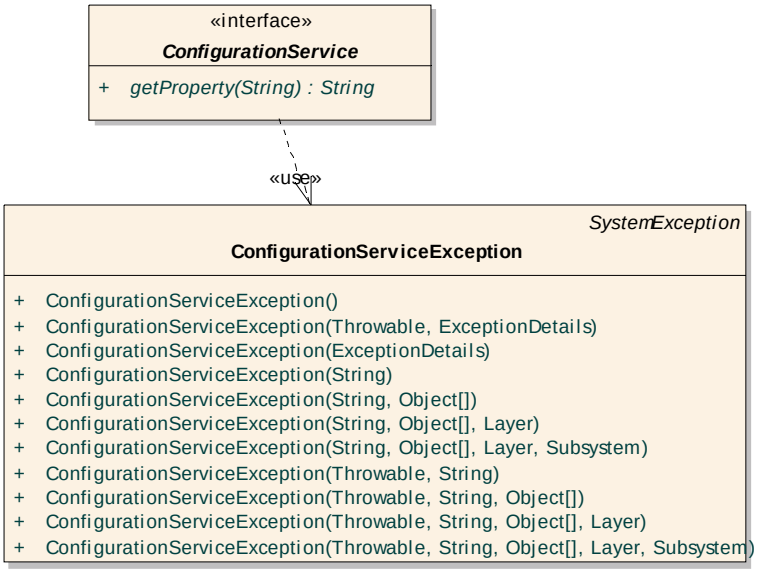
# 2. Descripció Detallada

## 2.1. Arquitectura i Components

Els components podem classificar-los en:

- Interfícies i Components Genèrics. Interfícies del servei i components d'ús general amb independència de la implementació escollida.
- Implementació basada en Spring

### *2.1.1. Interfícies i Components Genèrics*



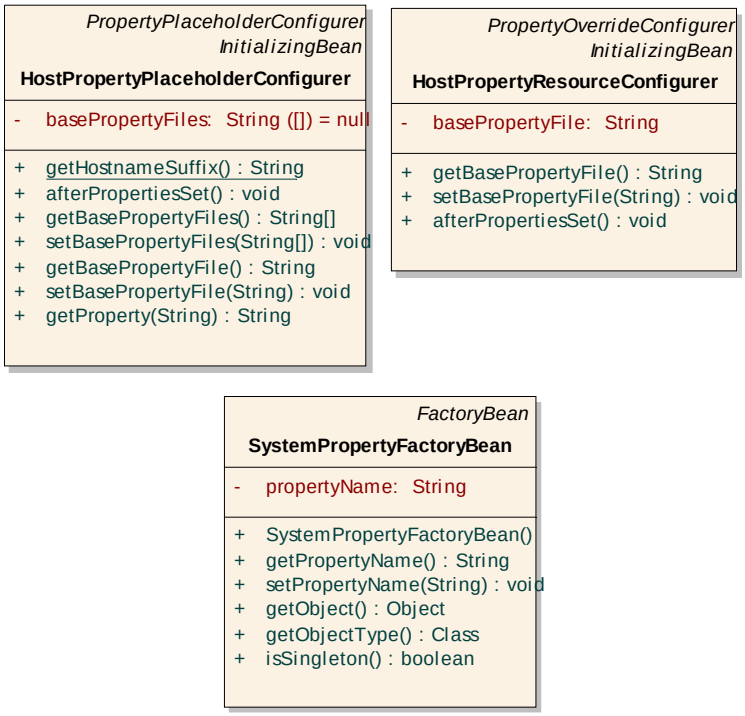

| Component | Package | Descripció |
|---|---|---|
| ConfigurationService | net.opentrends.openframe.services .configuration | Permet l'obtenció d'una propietat de l'aplicació mitjançant el mètode 'getProperty' |
| ConfigurationService Exception | net.opentrends.openframe.services .configuration.exception | Excepció que pot llençar el servei de configuració |

Es recomana no fer ús de forma directa d'aquest servei sinó usar el patró de 'Dependency Injection' pel qual totes les propietats s'injectaran automàticament.

### *2.1.2. Components basats en Spring*

A més de les classes pròpies de configuració que ofereix Spring, openFrame ofereix els següents components adicionals:

| Component | Package | Descripció |
|---|---|---|
| HostPropertyPlaceholderConfigurer | net.opentrends.openframe.services.configuration.springframework.beans.factory.config | Permet definir valors de configuració en un fitxer de propietats. Aquestes propietats estan identificades per un parell clau-valor a on el nom de la clau pot ser qualsevol identificador.<br><br>Extén la classe de Spring 'PropertyPlaceHolderConfigurer' per permetre definir varis fitxers de configuració segons el host en el que es trobi l'aplicació instal·lada<br><br>També és una implementació de la interfície 'ConfigurationService'. |
| HostPropertyResourceConfigurer | net.opentrends.openframe.services.configuration.springframework.beans.factory.config | **Permet sobreescriure amb valors de configuració de propietats de beans existents en un fitxer de propietats. Aquests valors de configuració estan identificats per un parell clau-valor a on el nom de la clau** es correspon **amb el nom d'una propietat d'un bean que existeix.**<br><br>**Extén la classe 'PropertyOverrideConfigurer' per permetre definir un fitxer de configuració que depén del host en el que es trobi l'aplicació instal·lada** |
| SystemPropertyFactoryBean | net.opentrends.openframe.services.configuration.springframework.beans.factory.systemproperty | Permet obtenir variables definides a l'entorn del sistema (Windows,Unix,...) |




Tant '*HostPropertyPlaceholderConfigurer*' com '*HostPropertyResourceConfigurer*' tenen la característica que el fitxer de propietats amb els valors de configuració depenen del host, és a dir, si li diem que volem carregar la configuració d'un fitxer que es diu config.properties, en realitat anirà a buscar un config.properties.nom_del_host.

Veure la documentació disponible al Javadoc per a més referència.

En l'apartat 'Configuració i Instal·lació' veurem en més detall la configuració d'aquests components.

## 2.2. Instal·lació i Configuració

### *2.2.1. Instal·lació*

La instal·lació del servei requereix de la utilització de la llibreria 'openFrame-services-configuration' i les dependències indicades a l'apartat 'Introducció-Versions i Dependències'.

### *2.2.2. Configuració*

La configuració del Servei de Configuració es realitza principalment amb el mecanisme d'injecció utilitzat. En el cas de Spring consultar 'http://static.springframework.org/spring/docs/1.2.x/reference/beans.html'.

A més de la configuració ja existent mitjançant el mecanisme d'injecció base (Spring) s'ofereixen les següents facilitats:

1) Definir un configurador que obtingui diferents valors de propietats segons la màquina en la que instal·lem l'aplicatiu

2) Definir un configurador per obtenir variables de l'entorn del sistema

- Consideració amb Spring

  En cap lloc s'especificarà quin és el servei de configuració que es fa servir per l'obtenció dels components i la seva injecció. Spring, per defecte cercarà totes les configuracions que s'hagin definit amb les classes configuradores (PropertyPlaceHolderConfigurer,...).

Configuració segons màquina

```
<bean id="configurationService"
      ...>
```

Atributs:

| Atributs | Requerit | Descripció |
|---|---|---|
| class | Sí | Implementació concreta del configurador a utilitzar<br><br>Usar el valor:<br>• net.opentrends.openframe.services.configuration.springframework.beans.factory.config.HostPropertyPlaceholderConfigurer<br><br>Si es volen definir varis fitxers de configuració<br><br>• net.opentrends.openframe.services.configuration.springframework.beans.factory.config.HostPropertyResourceConfigurer<br><br>Si es vol definir un únic fitxer de configuració |

- HostPropertyResourceConfigurer

  Propietats:

| Propietat | Requerit | Descripció |
|---|---|---|
| basePropertyFile | Sí | Fitxer de configuració amb la definició dels beans i les seves propietats<br>Per obtenir el fitxer de configuració s'obté el path definit en aquesta propietat i se li afegeix com a sufix el DNS de la màquina. És a dir, si s'ha configurat el fitxer 'nomFitxer.ext' es cercarà el fitxer 'nomFitxer.ext.nomHost'. |

- HostPropertyPlaceholderConfigurer

  Propietats:

| Propietat | Requerit | Descripció |
|---|---|---|
| basePropertyFiles | Sí | Llista de fitxers de configuració amb la definició dels beans i les seves propietats<br><br>Per obtenir el fitxer de configuració s'obté el path definit en aquesta propietat i se li afegeix com a sufix el DNS de la màquina. És a dir, si s'ha configurat el fitxer 'nomFitxer.ext' es cercarà el fitxer 'nomFitxer.ext.nomHost'. |

  Exemple:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<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>
                 </list>
          </property>
   </bean>
</beans>
```

  En el cas a dalt mostrat, si el DNS de la màquina és 'ES-57PYM1J' es cercarà el fitxer de configuració 'jdbc/jdbc.properties.ES-57PYM1J'. En cas que no es trobi aquest fitxer es cercaria el fitxer ' jdbc/jdbc.properties'.

Configuració de variables del sistema

```
<bean id="configurationService"
      ...>
```

TO DO

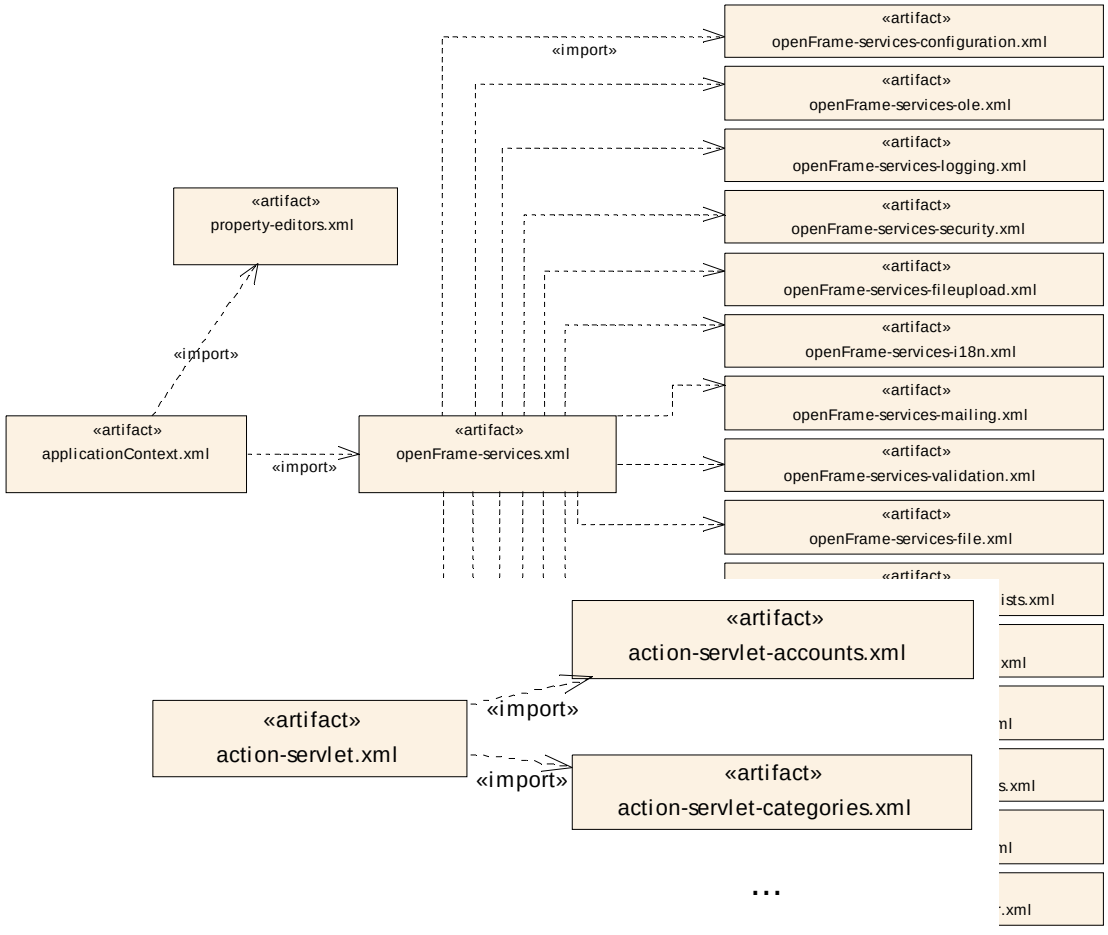
### 2.2.3. Ús de múltiples fitxers de configuració

Sovint és útil dividir el conjunt de definicions en múltiples fitxers XML. Mitjançant alguns dels configuradors és possible referenciar més d'un fitxer.

Tot i això és possible que volguem repartir els fitxers en diferents parts. En aquest cas podem fer ús de l'element 'import'.

```
<beans>
<import resource="services.xml"/>
<import resource="resources/messageSource.xml"/>
<import resource="/resources/themeSource.xml"/>
<bean id="bean1" class="..."/>
<bean id="bean2" class="..."/>
. . .
```

En aquest exemple, les definicions externes de beans s'estan carregant des de 3 fitxers, `services.xml`, `messageSource.xml` i `themeSource.xml`. Tots els paths són considerats relatius al fitxer de definició que realitza la importació, per tant en aquest cas `services.xml` ha d'estar en el mateix directori o classpath location que el fitxer que realitza la importació, mentres que `messageSource.xml` i `themeSource.xml` han d'estar en el directori resources per sota del fitxer que realitza la importació. Es pot comprovar que una barra inicial és ignorada, però atés que aquests paths són relatius, probablement és millor no usar la barra. Els continguts dels fitxers importats han de ser definicions XML de beans completament vàlides d'acord amb el DTD, incloent l'element bean de més alt nivell.



## 2.3. Utilització del Servei

Degut a que la configuració dels elements de l'aplicació o dels serveis es realitza de forma externa mitjançant el patró 'Dependency Injection' no és convenient l'obtenció de propietats des dels clients. Aquesta obtenció es realitza de forma transparent fora de les classes de l'aplicació.

### *2.3.1. Obtenció de valors de configuració des de les classes*

Si en algun cas es requereix de l'obtenció de propietats externes des del nostre codi (opció no recomanada), podem fer ús de la interfície 'ConfigurationService'.

```
package net.opentrends.openframe.services.configuration;

...
public interface ConfigurationService {
    public String getProperty(String key) throws
         ConfigurationServiceException;
}
```

Es proporciona una implementació basada en Spring, anomenada 'HostPropertyPlaceholderConfigurer'

El mètode que s'exposa és un i es correspon a la següent signatura:

- *`public String getProperty(String key)`*: Demana un valor de configuració amb la clau *key*.

Si el valor de configuració no es troba, es llença una excepció de tipus 'ConfigurationServiceException' indicant que el valor que es busca no s'ha trobat.

Per a utilitzar-lo, només cal que afegim al fitxer XML de definició de beans el nostre servei de configuració i injectem als nostres beans el servei de configuració, tal i com es mostra a continuació:

```
...
<bean id="myBean"
class="com.myapp.model.MyBean">
   <property name="configService" ref="configurationService" ❶/>
</bean>
...
<bean id="configurationService"
class="net.opentrends.openframe.services.configuration.springframework.beans.factory.config.HostPropertyResourceConfigurer❷">
   <property name="basePropertyFiles">
          <list>
                 <value>classpath:config.properties❸</value>
          </list>
   </property>
</bean>
...
```

❶ Referència al servei de configuració des del nostre bean
❷ Implementació del servei de configuració escollida
❸ Path del fitxer de propietats

I des de la classe que vol fer ús del servei:

```
...
public class MyBean {
  private ConfigurationService configService = null;

  public String myMethod() throws
  Exception {
        String configValue =
                    configService.getProperty("config.value");
        ...
  }
}
```

## 2.4. Eines de Suport

### *2.4.1. Spring IDE*

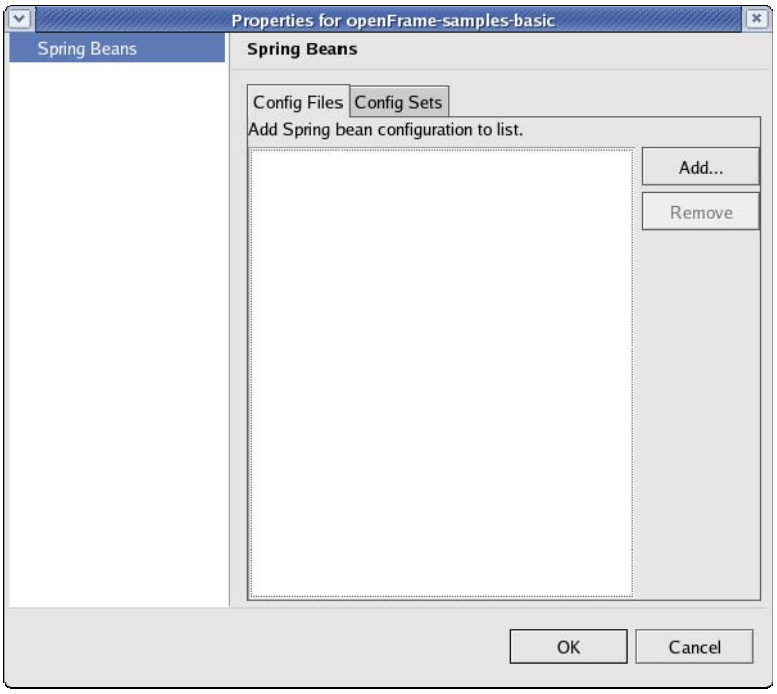
Instal·lació

Mitjançant l'opció de Eclipse 'Help|Software Updates|Find and Install...' introduir la url 'http://springide.org/project/wiki/SpringideInstall'

Comprovació de la Instal·lació

Per comprovar que s'ha instal·lat correctament seguir els següents pasos:

1) Seleccionar el node del projecte al navegador d'Eclipse i amb click dret seleccionar 'Add Spring Project Nature'.
2) Seleccionar 'Window|Show View|Other|Spring IDE|Spring Beans'
3) Amb botó dret seleccionar 'Properties' i apareixerà una finestra com la mostrada a continuació:




4) Prémer el botó 'Add' i seleccionar un fitxer de configuració de Spring. Ja podem començar a utilitzar el Plugin.

Per a més referència consultar el tutorial a la url 'http://springide.org/project/wiki/SpringideGuide'.

TO DO: Continuar...

## 2.5. Integració amb Altres Serveis

El Servei de Configuració és usat per tots els Serveis per tal de definir les seves propietats de forma externa.

En comptats casos, algunes de les implementacions sobre el que es troben els serveis de openFrame no permeten la definició de les seves propietats mitjançant la injecció.

### *2.5.1. Servei de Traces basat en Log4J*

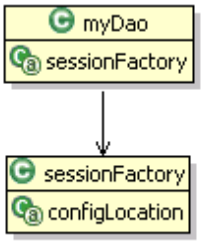
En el Servei de Traces openFrame ofereix la possibilitat de configurar diferents fitxers segons el host en el que es troba l'aplicació mitjançant la classe 'net.opentrends.openframe.services.logging.log4j.xml.HostDOMConfigurator'. Per a més referència consultar el document 'Servei de Traces'.

## 2.6. Preguntes Freqüents

# 3. Exemples

## 3.1. Exemple de Configuració

Imaginem que tenim un DAO que fa ús d'una factoria de sessions per llençar els objectes de negoci contra una capa de persistència i persistir les dades. Aquesta relació es representa en la següent figura:




Aquests elements i relacions es representen amb un fitxer XML amb una estructura definida. En el nostre exemple, tindríem un fitxer XML amb la següent informació:

- Element 'myDao'
- Element 'sessionFactory'
- Relació entre 'myDao' i 'sessionFactory'

A continuació es mostra aquest fitxer:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
   <!-- Element estructural que representa la factoria de sessions -->
   <bean name="sessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
   <property name="configLocation" value="classpath:hibernate.cfg.xml" />
   </bean>
   <!-- Element estructural que representa el DAO -->
   <bean id="myDao" class="test.dao.hibernate.impl.HibernateCategoryDAOImpl">
          <!-- Relació entre el nostre DAO i la factoria de sessions -->
          <property name="sessionFactory" ref="sessionFactory" />
   </bean>
</beans>
```

Tal i com defineix Spring, cada 'bean' representa un element o actor del nostre aplicatiu i es representa al XML com un node <*bean*/>. Els valors més importants a configurar en aquest node són:

- Atribut 'name': nom amb el que identifiquem el 'bean'
- Atribut 'class': nom de la classe que s'instanciarà quan es pregunti o resolgui el bean.

- Atributs de comportament: indiquen cóm ha de comportar-se el 'bean' en el contenidor. En l'exemple presentat es veu un cas molt senzill, en el que els elements representats són instàncies úniques en l'aplicatiu. Això i altres aspectes són completament configurables però s'escapen de l'objectiu d'aquest document (per exemple, si volem configurar que cada vegada que es demani un 'bean' pel 'name' es retorni una instància diferent, es farà servir l'atribut '**singleton**'). També existeixen atributs per indicar mètodes d'inicialització, '**init-method**' i destrucció '**destroy-method**').

Per a una explicació en més detall dels diferents atributs del node <*bean*/> consultar la url: '*http://static.springframework.org/spring/docs/1.2.x/reference/beans.html*'.

A part els atributs del node <*bean*/>, aquest pot tenir subnodes, que principalment representen altres 'beans' que necessita per a treballar, és a dir, els col.laboradors. En el nostre exemple hem definit un node <*property*/> amb el nom 'sessionFactory' que fa referència al col.laborador: el 'bean' amb nom 'sessionFactory' que representa la factoria de sessions..

El nom de les <*property*/> es correspon amb els noms de les variables d'instància del propi 'bean'

La definició d'aquests fitxers de configuració XML servirà posteriorment per anar a buscar en l'aplicatiu, en el cas que sigui necessari, les instàncies dels beans corresponents. Hi haurà doncs un procès automàtic que llegirà aquests fitxers i farà de punt d'entrada per a qualsevol petició de 'bean'. Aquest procès l'executa una entitat que es coneix com a 'BeanFactory'. L'existència d'aquesta factory en una aplicació J2EE bé representada per una classe que es diu 'WebApplicationContext'.

***Nota:***

L'existència d'aquesta factory és totalment transparent al desenvolupador, ja que no s'ha de preocuperar de conèixer, a no ser que sigui necessari, de l'existència de la mateixa, si no de la correcta configuració dels XML per a que aquesta resolgui les dependències entre col.laboradors correctament.

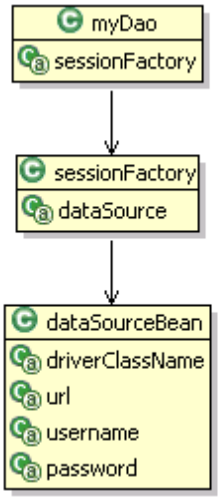
## 3.2. Exemple de Configuració amb Definició segons l'Entorn

En l'anterior exemple hem vist la forma de configurar la nostra aplicació en funció dels fitxers XML que representen els nostres 'beans' i les seves relacions. Però hi han aspectes que s'escapen en aquesta configuració:

1. Definició de propietats de 'beans' segons l'entorn de l'aplicació
2. Obtenció de valors de configuració que no pertànyin a cap bean

### 3.2.1. Definició de Propietats segons Entorn

Imaginem que en l'escenari que hem presentat a la Figura 1 i afegim un 'dataSourceBean' que fa de factoria de connexions i que té una relació de col.laboració amb la factoria de sessions:




Al fitxer de configuració abans presentat s'afegeix el següent bean:

```
...
<bean id="dataSourceBean"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
   <property name="driverClassName"
                  value="${jdbc.driverClassName}" />
   <property name="url" value="${jdbc.url}" />
   <property name="username" value="${jdbc.username}" />
   <property name="password" value="${jdbc.password}" />
</bean>
...
```

i es modifica la configuració del bean 'sessionFactory' per afegir la relació de col.laboració:

```
<property name="dataSource" ref="dataSourceBean" />
```

Si ens fixem en les propietats del bean 'dataSourceBean' veurem que hi han quatre *<property/>* sensibles de canviar segons l'entorn:

- driverClassName
- url
- username
- password

Els valors d'aquests atributs s'extreuen d'un fitxer extern segons la definició '${jdbc.driverClassName},${jdbc.url},${jdbc.username} i ${jdbc.password}. Però, d'ón es treuen els valors d'aquestes variables?

En aquest punt intervé el servei de configuració: mitjançant les classes 'HostPropertyPlaceholderConfigurer' i 'HostPropertyResourceConfigurer' que trobem en el package 'net.opentrends.openframe.services.configuration.springframework.beans.factory.config' es configura quina és la localització del fitxer extern.

```
...
<bean id="dataSourceBean"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
   <property name="driverClassName"
          value="${jdbc.driverClassName}" />
   <property name="url" value="${jdbc.url}" />
   <property name="username" value="${jdbc.username}" />
   <property name="password" value="${jdbc.password}" />
</bean>
...
<bean id="configurationService"
class="net.opentrends.openframe.services.configuration.springframework.beans.factory.config.HostPropertyPlaceholderConfigurer">
   <property name="basePropertyFiles">
          <list>
                 <value>classpath:config.properties❶</value>
          </list>
   </property>
</bean>
...
```

❶ Configuració del fitxer extern

Aquest fitxer extern defineix els valors de les propietats:

```
jdbc.driverClassName=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:d:/path_to_db/testDB
jdbc.username=sa
jdbc.password=
```

# 4. Annexos