Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Servei de Seguretat

Versió 1.0

Barcelona, 21 / febrer / 2006

Històric de modificacions

Data	Autor	Comentaris	Versió
05/01/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0

Llegenda de Marcadors

Índex

1. Introducció

1.1. Propòsit

El Servei de Seguretat té com a propòsit principal gestionar l'autentificació i l'autorització dels usuaris de les nostres aplicacions. L'objectiu de l'autentificació és comprovar que l'usuari és qui diu ser, mentre que l'autorització s'encarrega de comprovar que realment té accés al recurs sol·licitat.

NOTA:

L'especificació JAAS (Java Authorization and Authentication) de J2EE proporciona els mecanismes necessaris de seguretat. Cada servidor d'aplicacions pot implementar l'estàndard però ho fa de diferents formes produint problemes de compatibilitat. L'especificació JAAS s'orienta principalment a temes d'autentificació, mentre que els temes d'autorització pateixen de moltes carències.

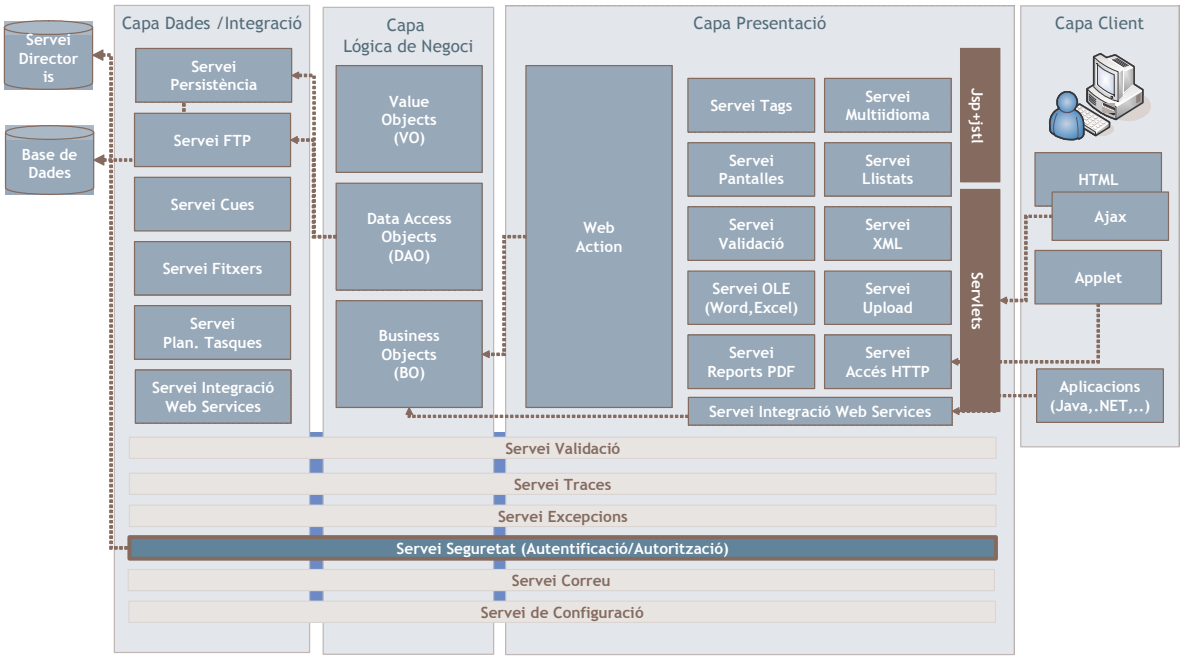
Actualment, i a causa del seu grau de maduresa i facilitat openFrame recomana l'ús de 'Acegi' com framework base i les extensions que openFrame proporciona.

Per a l'ús d'aquest servei i la lectura del present document es necessiten com a prerequisits els següens aspectes:

1) Coneixements sobre bases de dades
2) Coneixements bàsics sobre LDAP
3) Coneixements bàsics sobre Filtres Servlet i llibreries de tags JSP
4) Coneixements bàsics amb Spring

1.2. Context i Escenari d'ús

El Servei de Seguretat és un servei general d'openFrame.



1.3. Versions i Dependències

En el present apartat es mostren les versions i dependències necessàries per a l'ús del Servei.

Nom	Tipus	Versió	Descripció
servlet-api	jar	2.4	
jsp-api	jar	2.0	http://java.sun.com/products/jsp
acegi-security	jar	0.8.3	http://acegisecurity.sourceforge.net
spring	jar	1.2.5	http://www.springframework.org
commons-collections	jar	3.1	http://jakarta.apache.org/commons
commons-codec	jar	1.3	http://jakarta.apache.org/commons
commons-lang	jar	2.1	http://jakarta.apache.org/commons
ehcache	jar	1.1	http://ehcache.sourceforge.net
oro	jar	2.0.8	
jstl	jar	1.0.6	http://jakarta.apache.org/taglibs/
standard	jar	1.0.6	http://jakarta.apache.org/taglibs/
xercesImpl	jar	2.6.2	
xml-apis	jar	2.0.2	http://xml.apache.org/commons/

1.4. A qui es dirigeix

Aquest document es dirigeix als perfils següents:

- Arquitecte. Per a definir una estratègia de seguretat i la seua implementació amb Acegi
- Programador. Per a configurar el servei de seguretat de cada aplicació web

1.5. Documents i Fonts de Referència

Guió de referència D'Acegi	http://acegisecurity.sourceforge.net/reference.html
Introducció d'Acegi	http://javahispano.net/frs/download.php/120/SeguridadnointrusivaSpring.pdf

1.6. Glossari

ACLs (Access Control List)

Les llistes de control d'accés o ACLs permeten gestionar les autoritzacions de cada rol o usuari a nivell de dades o instàncies de lògica de negoci. Per a cada instància es poden especificar diferents dret: lectura, escriptura, esborrat, creació.

2. Descripció Detallada

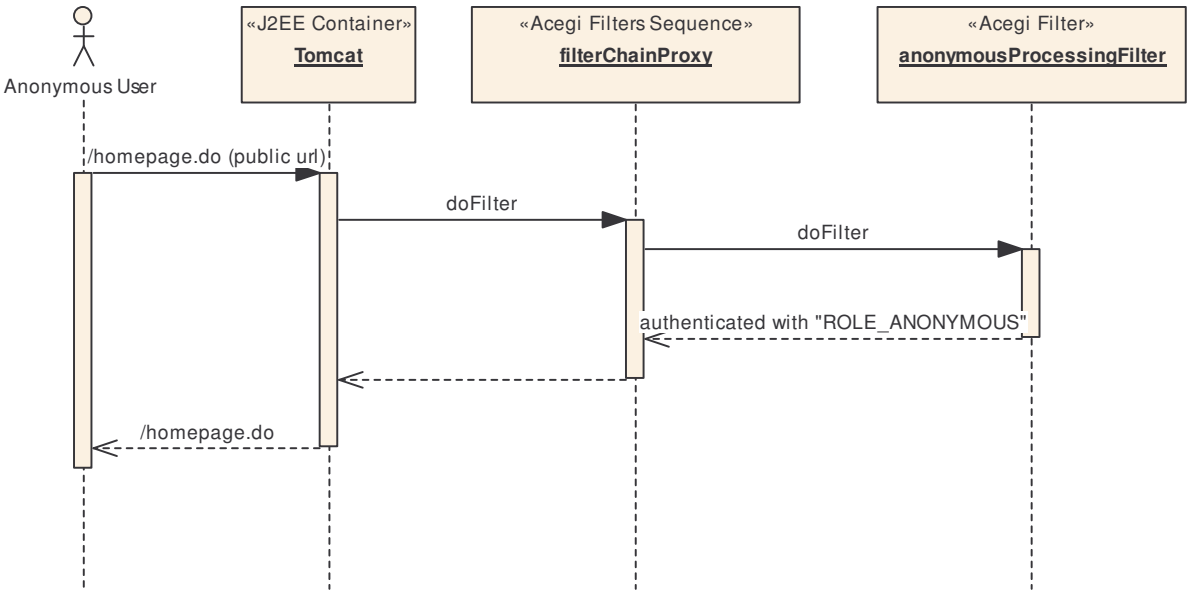
2.1. Arquitectura Bàsica

En el present apartat es mostren els diagrames de seqüència associats al tractament de peticions i cóm el servei de seguretat activarà el pas o no al recurs sol·licitat.

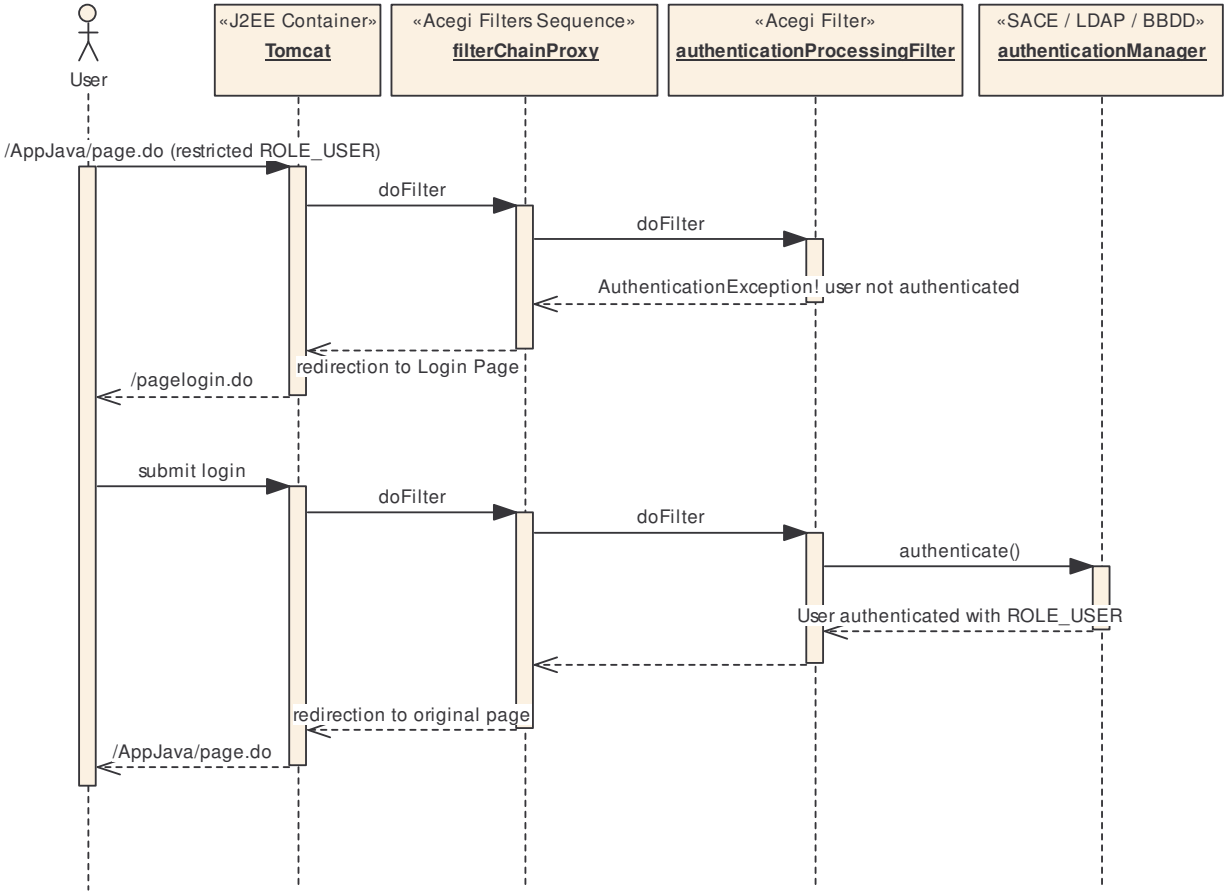
2.1.1. Accés a una url no protegida

Si hi ha un accés a una url que no s'ha configurat com a protegida (veure apartat 'Configuració') es segueixen els següents pasos:

1) S'activen uns filtres que comproven si l'usuari està autentificat
2) Donat que la url no és protegdia, el servei tramita la petició amb el filtre "anonymousProcessingFilter" i assigna a la sessió l'usuari 'ANONYMOUS' i el rol 'ROLE_ANONYMOUS'
3) L'usuari a partir d'aquest moment es troba autentificat i pot accedir a totes les urls no protegides



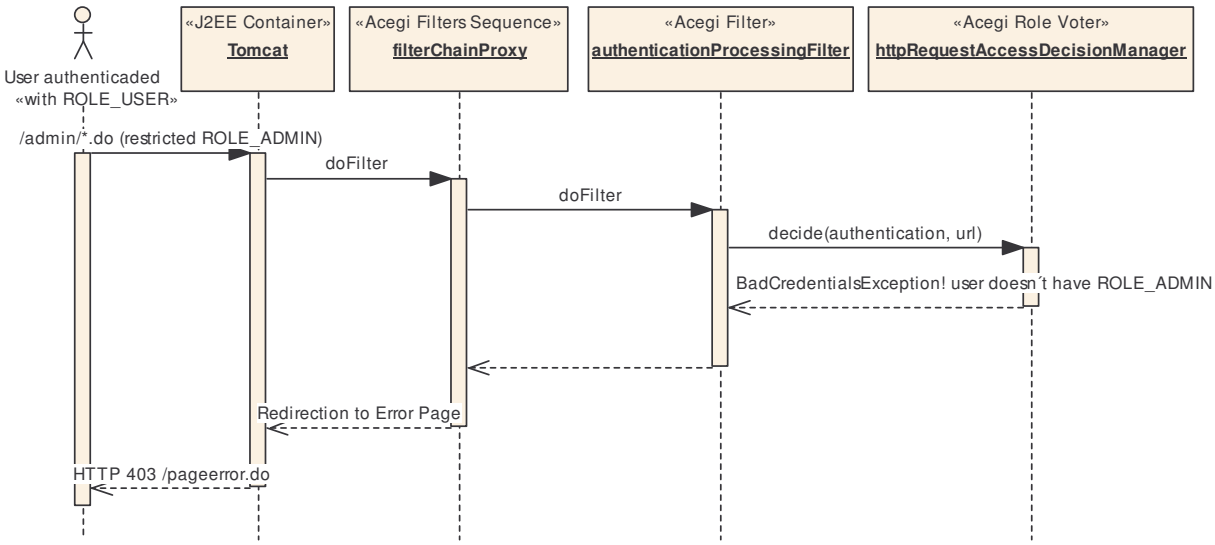
2.1.2. Accés a urls protegides



Si s'accedeix a una url protegida, el funcionament és:

1) El servei executa una seqüència de filtres per comprovar l'autentificació.
2) Com la url és protegida, el servei gestiona la petició amb el filtre "authenticationProcessingFilter" y llença una excepció de tipus AuthenticationException (usuari no autentficat).
3) Es redirecciona l'usuari a la pàgina de Login.
4) De nou, s'executen uns filtres per tramitar la petició d'autentificació
5) Es crida al mètode 'authenticate()'. El authenticationManager comprova els noms d'usuaris /password contra una seqüència d'autentificació (SACE / LDAP / BBDD).
6) Finalment l'usuari es troba autentificat (i configurat amb el seu(s) rol(s)) i se li redirigeix a la pàgina inicialment sol·licitada

2.1.3. Accés a urls protegides si usuari autentificat

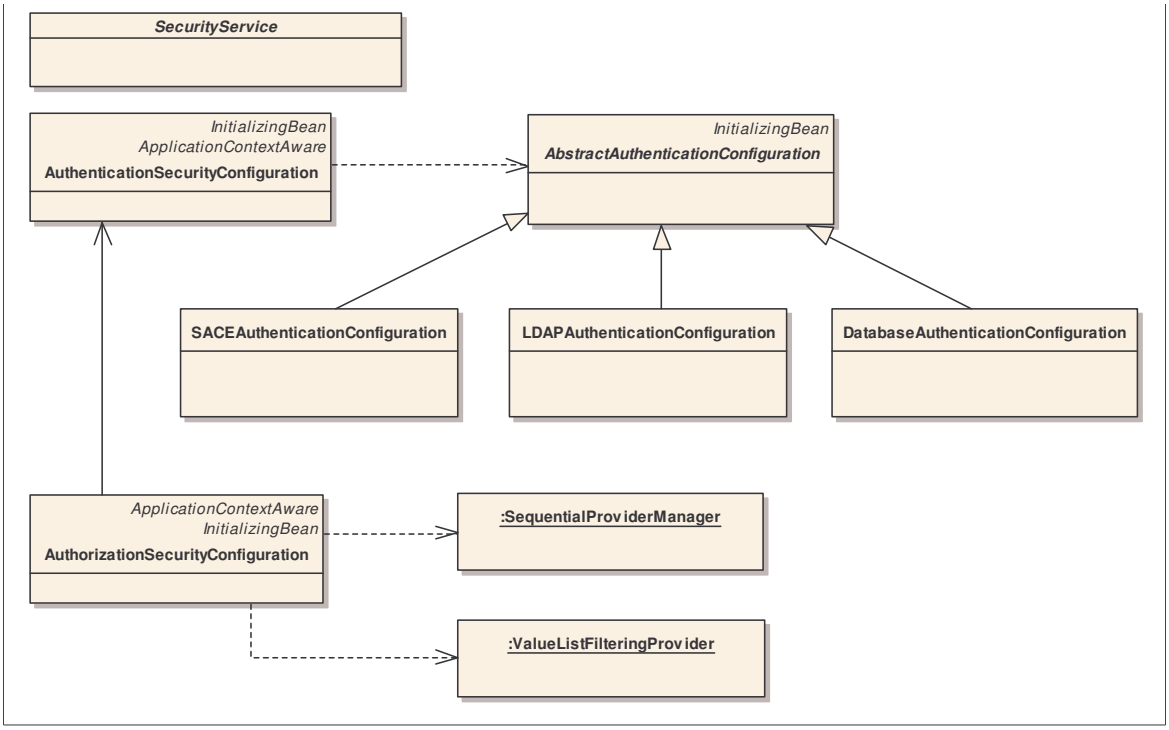


Si el mateix usuari amb un rol 1 intenta accedir a una url restringida pel rol 2, la seqüència és:

1) Donat que la url és protegida, es tramita la petició amb el filtre "authenticationProcessingFilter" per comprovar que l'usuari està autentificat.
2) Es fa un crida al mètode decide() dels voters configurats. Com la url és protegida pel rol 2 (per exemple 'ROLE_USER') i l'usuari només té el rol 2 (per exemple 'ROLE_ADMIN'), es llença una excepció de tipus BadCredentialsException
3) Se li redirigeix a una pàgina de HTTP 403 d'error (es configurar en el fitxer web.xml)

2.2. Interfícies i Components Genèrics

2.2.1. Interfícies Principals



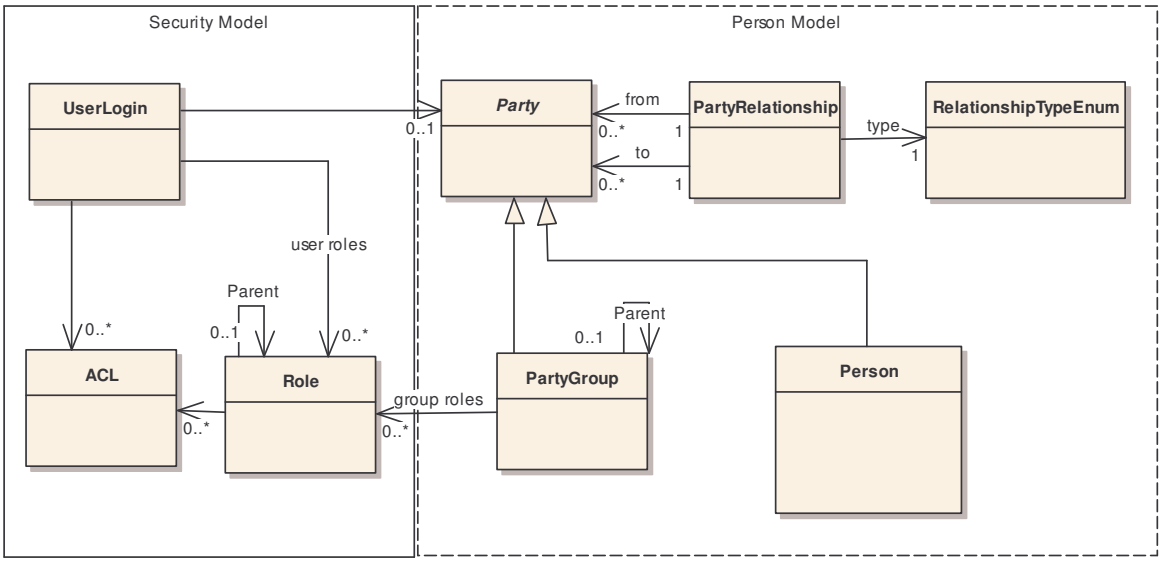
Només la interfície 'SecurityService' serà visible des de les classes desenvolupades per les aplicacions, les restants classes s'usaran per definir mitjançant la configuració el comportament dessitjat per la seguretat.

Component	Package	Descripció
SecurityService	net.opentrends .openframe .services .security	Proporciona mètodes d'accés a informació de context de seguretat (veure si l'usuari està autenticat, és d'un rol determinat, etc.)
AuthenticationSecurityConfiguration	net.opentrends .openframe .services .security	Classe de configuració de l'autentificació
AbstractAuthenticationConfiguration	net.opentrends .openframe .services .security	Superclasse de les classes de configuració de l'autentificació dels usuaris per base de dades, LDAP, etc ...
SACEAuthenticationConfiguration	net.opentrends .openframe	Classe de configuració de l'autentificació per SACE

Component	Package	Descripció
	.services .security	
DatabaseAuthenticationConfiguration	net.opentrends .openframe .services .security	Classe de configuració de l'autentificació per base de dades
LDAPAuthenticationConfiguration	net.opentrends .openframe .services .security	Classe de configuració de l'autentificació per LDAP
AuthorizationSecurityConfiguration	net.opentrends .openframe .services .security	Classe de configuració de l'autorització
SequentialProviderManager	net.opentrends .openframe .services .security.acegi	Classe que permet una autentificació amb una o diverses fonts de dades (BBDD, LDAP) en seqüència
ValueListFilteringProvider	net.opentrends .openframe .services .security.acegi	Classe que filtra les llistes d'instància i elimina aquelles instàncies a què l'usuari no pot accedir pels seus permisos.

2.2.2. Model de Classes de Seguretat

El model a continuació mostrat és el model de representació de la informació de la seguretat en relació al model de persones, rols, usuaris i grups.



Clase	Package	Descripció
Party	net.opentrends .openframe .services .security.model	Implementació del patró de disseny 'Party' per representar les relacions entre persones i organitzacions (veure http://www.tdan.com/i021ht04.htm per més referència)
PartyRelationship	net.opentrends .openframe .services .security.model	Asociació entre 2 "Party" (Group o User)
PartyRelationshipTypeEnum	net.opentrends .openframe .services .security.model	Tipus de la relació.
PartyGroup	net.opentrends .openframe .services .security.model	Grup d'usuaris. Pot tenir associats un o més rols i un grup pot també pertànyer a un grup pare.
Role	net.opentrends .openframe .servicesl	Representació de Rols i subrols.

Clase	Package	Descripció
	.security.model	
UserLogin	net.opentrends .openframe .services .security.model	Dades de l'usuari de les aplicacions. Serà la informació d'entrada a les aplicacions associada a un usuari o grup
ACL	net.opentrends .openframe .services .security.model	Autoritzacions a nivell d'instància de la lógica de negoci.

2.3. Instal·lació i Configuració

Prerequisit: Per al funcionament correcte d'aquest servei és important haver realitzat prèviament les configuracions especificades de la part 'Configuració Bàsica' del document 'Servei de Presentació'.

Per a la instal·lació del Servei de Seguretat necessitem usar el fitxer 'openFrame-services-security.jar' i les dependències indicades en l'apartat 'Versions i Dependències'.

La configuració del servei de seguretat es descompon en les parts següents:

- Configuració de la Base de Dades
- Configuració dels Filtres de l'Aplicació Web
- Configuració de la Font de Dades de l'Esquema de Seguretat
- Configuració de l'Autentificació
- Configuració de l'Autorització
- Configuració del Servei d'Accés a la Informació de l'Usuari. Aquesta configuració ens permetrà fer ús d'una interfície per obtenir dades de l'usuari connectat (si és d'un rol determinat, ...)

El Servei de Seguretat d'openFrame encapsula la complexitat de "Acegi" per mitjà d'una configuració simplificada i centralitzada en dos beans: authenticationConfiguration i authorizationConfiguration.

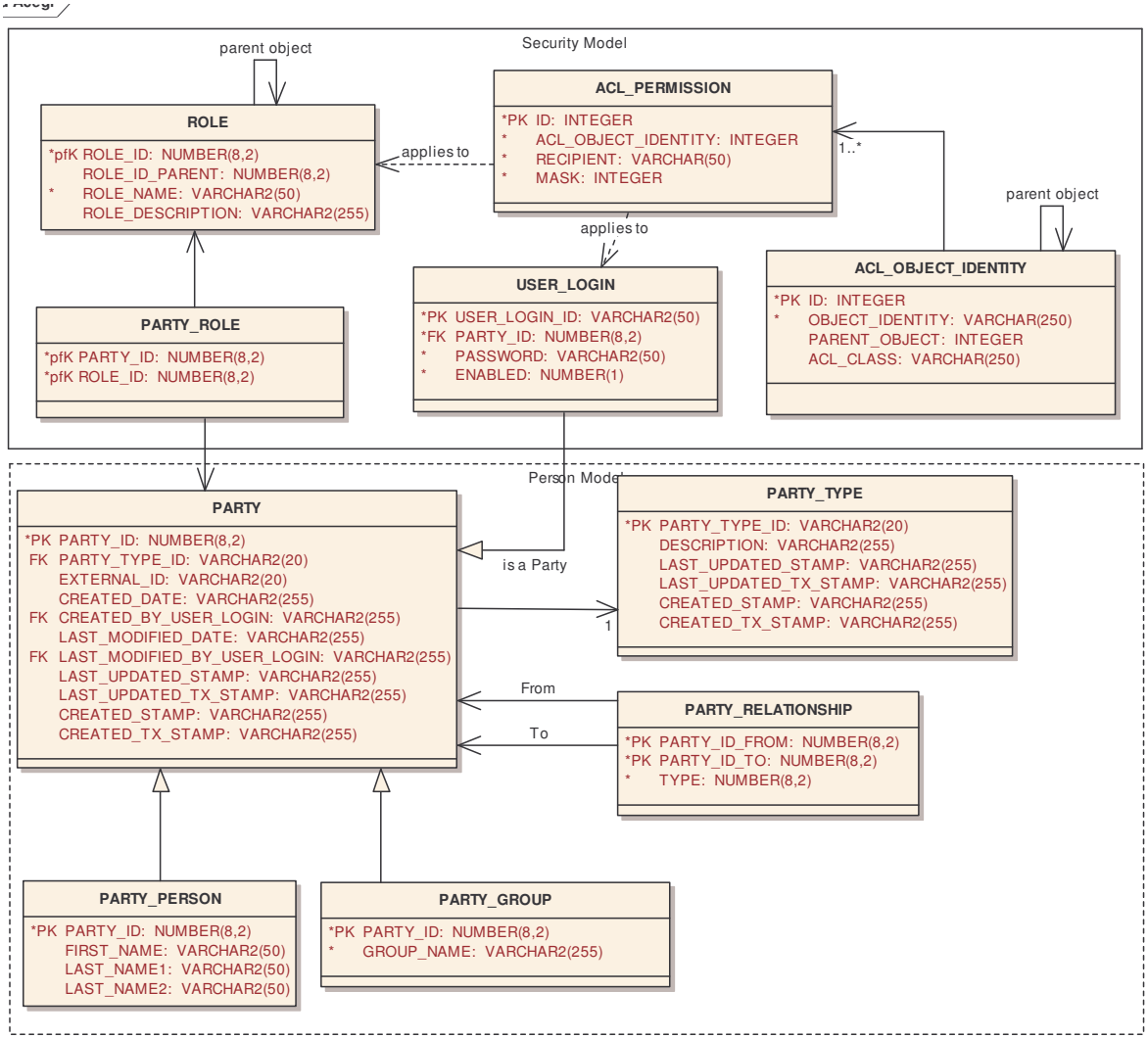
NOTA prèvia:
Acegi **no funciona** si s'usen caràcters de retorn en les definicions de les propietats de tipus "<list>" per als beans de seguretat.

Exemple:

Per a més informació sobre el framework "Acegi", consultar la pàgina 'http://acegisecurity.sourceforge.net/suggested.html'

2.3.1. Configuració de la Base de Dades

Per l'ús de la seguretat openFrame utilitza, per defecte, l'esquema de base de dades mostrat a continuació:



ⓘ NOTA:

Si la base de dades d'usuaris ja existeix en el moment d'implantació d'openFrame, no cal crear les taules d'autentificació 'USER_LOGIN', 'PARTY_ROLE' i 'ROLE'. Haurem d'indicar les query per a obtenir el nom d'usuari, password i rol.

Taules d'Autentificació

1) USER_LOGIN

Els camps de la taula "USER_LOGIN" són:

- "party_id": clau primària
- "username": clau primària, representa el nom de l'usuari.
- "password: representa la contrasenya, que pot ser codificada
- "enabled": Indica si l'usuari està habilitat o no

2) ROLE

Els camps de la taula "ROLE" són:

- "role_id": clau primària.
- "role_id_parent": clau forània, representa el rol pare (opcional)
- "role_name", nom del rol. El seu nom ha de començar amb el prefix "ROLE_". Exemple: ROLE_USUARIO, ROLE_GESTOR, ...
- "role_description", representa la descripció del rol.

3) PARTY_ROLE

Els camps de la taula "PARTY_ROLE" són:

- "username": clau primària, representa el nom de l'usuari
- "authority": nom del rol de l'usuari. El seu nom ha de començar amb el prefix "ROLE_". Exemple: ROLE_USUARIO, ROLE_GESTOR, ...

Taules d'Autorització

Les taules d'autorització només són necessàries si necessitem gestionar les autoritzacions a nivell d'instàncies de la lògica de negoci (ACLs).

1) ACL_OBJECT_IDENTITY

La taula "ACL_OBJECT_IDENTITY" ens permet identificar els objectes de la lògica de negoci. Els seus camps són:

- "id": identificador i clau primària
- "object_identity": identificador String de l'objecte. Es seguirà com a convenció 'nomClasse:id', on nomClasse correspon al nom 'full qualified' de la classe de negoci i id al valor de l'identificador de l'objecte. Exemple: "com.business.domain.DomainObject:1"
- "parent_object". Aquest camp permet especificar l'objecte pare de l'objecte actual. D'aquesta manera, en cas de no disposar d'autoritzacions (en la taula ACL_PERMISSION) associades a l'objecte s'usaran les autoritzacions que s'hagin definit per al seu pare.
- "acl_class". En aquest valor sempre haurem d'introduir "net.sf.acegisecurity.acl.*basic*.SimpleAclEntry"

2) ACL_PERMISSION

La taula ACL_PERMISSION conté els permisos que s'apliquin als usuaris o rols per a cada objecte de la lògica de negoci. Conté els camps següents:

- "id":identificador i clau primària.
- "acl_object_identity": clau forània, que referència a una fila de la taula 'ACL_OBJECT_IDENTITY'
- "recipient": Cadena que representa els noms d'usuari o noms de rols als què s'aplica el permís. Cada nom ha de ser separat per ','. Es poden representar un conjunt de noms i rols d'usuari de forma conjunta i barrejada.
- "mask": representa la màscara del permís que s'aplica a l'objecte. En la taula següent es presenta la llista de valors possibles:

Valor	Permisos
0	No té permisos a l'objecte
1	'ADMIN' (Permisos d'administració)
2	'READ' (Permís de lectura)
4	'WRITE' (Permís d'escriptura)

Valor	Permisos
6	‘READ + WRITE
8	‘CREATE' (Permís de creació)
14	‘READ + WRITE + CREATE'
16	‘DELETE' (Permís d'esborrat)
22	READ + WRITE + DELETE
30	READ + WRITE + CREATE + DELETE

Per a la creació de l'esquema podem fer ús del *script* següent per Oracle:

```
DROP SEQUENCE ACL_PERMISSION_SEQ;
DROP SEQUENCE ACL_OBJECT_SEQ;
DROP SEQUENCE PARTY_OBJECT_SEQ;
DROP SEQUENCE ROLE_OBJECT_SEQ;

create sequence ROLE_OBJECT_SEQ
start with 1
increment by 1;

create sequence ACL_OBJECT_SEQ
start with 1
increment by 1;

create sequence ACL_PERMISSION_SEQ
start with 1
increment by 1;

create sequence PARTY_OBJECT_SEQ
start with 1
increment by 1;


DROP TABLE ACL_PERMISSION;
DROP TABLE ACL_OBJECT_IDENTITY;
DROP TABLE PARTY CASCADE CONSTRAINTS;
DROP TABLE GROUP_ROLE CASCADE CONSTRAINTS;
DROP TABLE PARTY_GROUP CASCADE CONSTRAINTS;
DROP TABLE PARTY_RELATIONSHIP CASCADE CONSTRAINTS;
DROP TABLE PARTY_ROLE CASCADE CONSTRAINTS;
DROP TABLE ROLE CASCADE CONSTRAINTS;
DROP TABLE USER_LOGIN CASCADE CONSTRAINTS;

CREATE TABLE acl_object_identity (
     id NUMBER PRIMARY KEY,
     object_identity VARCHAR2(250) NOT NULL,
     parent_object INTEGER,
     acl_class VARCHAR2(250) NOT NULL,
     CONSTRAINT unique_object_identity UNIQUE(object_identity),
     FOREIGN KEY (parent_object) REFERENCES acl_object_identity(id)
);

CREATE TABLE acl_permission (
     id NUMBER PRIMARY KEY,
     acl_object_identity NUMBER NOT NULL,
     recipient VARCHAR2(100) NOT NULL,
     mask NUMBER NOT NULL,
     CONSTRAINT unique_recipient UNIQUE(acl_object_identity, recipient),
     FOREIGN KEY (acl_object_identity) REFERENCES acl_object_identity(id)
);
```

```
CREATE TABLE PARTY (
   PARTY_ID NUMBER NOT NULL,
   PARTY_TYPE_ID NUMBER,
   EXTERNAL_ID VARCHAR2(20),
   CREATED_DATE VARCHAR2(255),
   CREATED_BY_USER_LOGIN VARCHAR2(255),
   LAST_MODIFIED_DATE VARCHAR2(255),
   LAST_MODIFIED_BY_USER_LOGIN VARCHAR2(255),
   LAST_UPDATED_STAMP VARCHAR2(255),
   LAST_UPDATED_TX_STAMP VARCHAR2(255),
   CREATED_STAMP VARCHAR2(255),
   CREATED_TX_STAMP VARCHAR2(255),
   CONSTRAINT PK_PARTY_ID UNIQUE(PARTY_ID)
)
;

CREATE TABLE PARTY_GROUP (
   PARTY_ID NUMBER NOT NULL,
   GROUP_PARENT_ID NUMBER NULL,
   GROUP_NAME VARCHAR2(255) NOT NULL,
   CONSTRAINT PK_PARTY_GROUP UNIQUE(PARTY_ID)
)
;

CREATE TABLE PARTY_RELATIONSHIP (
   PARTY_ID_FROM NUMBER NOT NULL,
   PARTY_ID_TO NUMBER NOT NULL,
   TYPE NUMBER NULL,
   CONSTRAINT PK_PARTY_RELATIONSHIP UNIQUE(PARTY_ID_FROM, PARTY_ID_TO),
   FOREIGN KEY (PARTY_ID_FROM) REFERENCES PARTY(PARTY_ID),
   FOREIGN KEY (PARTY_ID_TO) REFERENCES PARTY(PARTY_ID)
)
;

CREATE TABLE ROLE (
   ROLE_ID NUMBER NOT NULL,
   ROLE_ID_PARENT NUMBER,
   ROLE_NAME VARCHAR2(50) NOT NULL,
   ROLE_DESCRIPTION VARCHAR2(255),
   CONSTRAINT PK_ROLE UNIQUE(ROLE_ID),
   FOREIGN KEY (ROLE_ID_PARENT) REFERENCES ROLE(ROLE_ID)
)
;

CREATE TABLE USER_LOGIN (
   USER_LOGIN_ID VARCHAR2(50) NOT NULL,
   PARTY_ID NUMBER NOT NULL,
   PASSWORD VARCHAR2(50) NOT NULL,
   CONSTRAINT PK_USER_LOGIN UNIQUE(USER_LOGIN_ID),
   FOREIGN KEY (PARTY_ID) REFERENCES PARTY_GROUP(PARTY_ID)
)
;

CREATE TABLE PARTY_ROLE (
   USER_LOGIN_ID VARCHAR2(50) NOT NULL,
   ROLE_ID NUMBER NOT NULL,
   CONSTRAINT PK_PARTY_ROLE UNIQUE(USER_LOGIN_ID, ROLE_ID),
   FOREIGN KEY (USER_LOGIN_ID) REFERENCES USER_LOGIN(USER_LOGIN_ID),
   FOREIGN KEY (ROLE_ID) REFERENCES ROLE(ROLE_ID)
)
;

CREATE TABLE GROUP_ROLE (
   PARTY_ID NUMBER NOT NULL,
   ROLE_ID NUMBER NOT NULL,
   CONSTRAINT PK_GROUP_ROLE UNIQUE(PARTY_ID, ROLE_ID),
```

```
   FOREIGN KEY (PARTY_ID) REFERENCES PARTY_GROUP(PARTY_ID),
   FOREIGN KEY (ROLE_ID) REFERENCES ROLE(ROLE_ID)
)
;

CREATE OR REPLACE TRIGGER ACL_OBJ_TRIGGER
before insert on ACL_OBJECT_IDENTITY
for each row
begin
select ACL_OBJECT_SEQ.nextval into :new.id from dual;
end;

CREATE OR REPLACE TRIGGER ACL_PERMISSION_TRIGGER
before insert on ACL_PERMISSION
for each row
begin
select ACL_PERMISSION_SEQ.nextval into :new.id from dual;
end;

CREATE OR REPLACE TRIGGER PARTY_OBJECT_SEQ_TRIGGER
before insert on PARTY_GROUP
for each row
WHEN (new.PARTY_ID is null)
begin
select PARTY_OBJECT_SEQ.nextval into :new.PARTY_ID from dual;
end;

CREATE OR REPLACE TRIGGER OPENFRAME.ROLE_OBJECT_SEQ_TRIGGER
before insert on ROLE
for each row
WHEN (new.ROLE_ID is null)
begin
select ROLE_OBJECT_SEQ.nextval into :new.ROLE_ID from dual;
end;
```

2.3.2. Configuració dels filtres de l'Aplicació Web

Fitxer de configuració: web.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/webapp/WEB-INF/web.xml

Acegi usa un conjunt de filtres per a detectar aspectes de l'autorització i autentificació. Per a usar-los definirem en el fitxer 'WEB-INF/web.xml el codi següent:

```
<!-- Filter ACEGI, using a FilterChainProxy makes it easier to configure
          different Filters in the Spring application context configuration file--
>
   <filter>
          <filter-name>Acegi Filter Chain Proxy</filter-name>
          <filter-class>
                 net.sf.acegisecurity.util.FilterToBeanProxy
          </filter-class>
          <init-param>
                 <param-name>targetClass</param-name>
                 <param-value>
                        net.sf.acegisecurity.util.FilterChainProxy
                 </param-value>
          </init-param>
   </filter>
```

```
    <filter-mapping>
           <filter-name>Acegi Filter Chain Proxy</filter-name>
           <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Per a més informació consultar la pàgina
http://acegisecurity.sourceforge.net/docbook/acegi.html#security-filters

2.3.3. Configuració de la font de dades de Seguretat

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

El Servei de Seguretat necessita conèixer com connectar-se a l'esquema de base de dades que defineix les taules anteriorment comentades. Per a això, hem d'especificar un bean 'dataSource' en el que configurarem les propietats de la base de dades. Es recomana l'ús de JNDI per a definir la font de dades.

Exemple amb jndi:

```
<bean     id="dataSource"
   class="org.springframework.jndi.JndiObjectFactoryBean">
   <property      name="jndiName"
          value="java:comp/env/JNDISource..."/>
</bean>
```

Exemple amb jdbc:

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

2.3.4. Configuració de l'Autentificació

En la configuració de l'Autentificació tindrem en consideració:

- Seleccionar la configuració de la font en que es realitza l'autentificació (per base de dades, per LDAP, per servei integrador al servidor corporatiu basat en HTTPS, ...)
- Configurar el formulari d'autentificació web i la seqüència de cerca on ha de realitzar-se l'autentificació (primer un LDAP, després una BD, etc.)

Configuració de la Font d'Autorització per Base de Dades

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

La configuració de l'autentificació per mitjà de base de dades es realitza per mitjà del bean de la classe '`net.opentrends.openframe.services.security.acegi.DatabaseAuthenticationConfiguration'`, per a la que podem definir les següents propietats:

Propietat	Requerit	Descripció
passwordEncoderClass	Sí	Aquesta propietat permet especificar quina encriptació ha de realitzar-se dels passwords. És a dir, si el password s'emmagatzema per exemple en la base de dades de forma encriptada, el servei de seguretat hauria d'encriptar el password introduït per l'usuari i comparar-ho amb el de la base de dades. La llista de possibles valors és: • BaseDigestPasswordEncoder, • BasePasswordEncoder, • Md5PasswordEncoder, • PlaintextPasswordEncoder, • ShaPasswordEncoder

Propietat	Requerit	Descripció
usersByUserNameQuery	No	Aquesta propietat permet especificar una query SQL per a l'obtenció d'un usuari. Serveix per a poder usar un esquema d'usuaris diferent al proposat en el present document (per exemple si ja existia abans de la implantació de openFrame). S'ha de seguir el següent patró: "SELECT {*Nom_usuario*}, {*contrassenya*} FROM {*usuaris*} WHERE {*Nom_usuario*} =? On Nom_usuario correspon al nom del camp de la taula que conté el nom de l'usuari, contrassenya, al nom del camp que conté el password i 'usuaris' al nom de la taula d'usuaris.
authoritiesbyUserNameQuery	No	Aquesta propietat permet especificar on es troba la informació dels rols. El seu ús ens permet usar un esquema de rols diferent al proposat en el present document. "SELECT *{Nom_usuario}*, *{rol}* FROM *{taula_rols}* WHERE {*Nom_usuario*} =?

Exemple:

```
<bean id="databaseAuthenticationConfiguration1"
          class="net.opentrends.openframe.services.security.
                                DatabaseAuthenticationConfiguration">


   <!-- Propietats obligatòries -->
   <property
          name   ="passwordEncoderClass"
   value  ="net.sf.acegisecurity.providers.encoding.
                                      PlaintextPasswordEncoder" />

   <!-- Propietats opcionals -->
   <property     name   ="usersByUserNameQuery">
          <value>
                        SELECT {Nombre_usuario}, {contraseña}
                FROM {usuarios}
                        WHERE {Nombre_usuario} =?
                </value>
   </property>


   <property     name   ="authoritiesbyUserNameQuery">
          <value>
```

```
                    SELECT {Nombre_usuario}, {rol}
                FROM {privilegios}
                    WHERE {Nombre_usuario} =?
                </value>
    </property>


    </bean>
```

Configuració de la Font d'Autorització per LDAP

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

En el cas que es requereixi configurar l'autentificació per mitjà de crides directes a LDAP podem fer ús d'una classe diferent.

Per a realitzar les proves en desenvolupament podem instal·lar un servidor LDAP senzill (veure l'apartat 'Eines de Suport' per a més referència).

La configuració de l'accés al LDAP es fa per mitjà d'un bean de la classe '`net.opentrends.openframe.services.security.acegi.LDAPAuthenticationConfiguration'`, per a la que podrem definir les propietats següents:

Propietat	Requerit	Descripció
ldapURL	Sí	Url del servidor LDAP. Per exemple: ldap://dir.mycompany.com:389/dc=mycompany,dc=com
usernameFormat	Sí	El patró del nom d'usuari usat pel LDAP. Per exemple, per a openLDAP: "uid={0},ou=people,dc=mycompany,dc=com"
userLookupNameFormat	Sí	El patró del nom d'usuari usat pel LDAP per a fer cerques. Per exemple, per a openLDAP: uid={0},ou=people

Exemple:

```
<bean    id="ldapAuthenticationConfiguration"
         class="net.opentrends.openframe.services.security.
                    LDAPAuthenticationConfiguration">

   <!-- Propietats obligatòries -->
   <property
```

```
        name   ="ldapURL"
    value  ="ldap://localhost:389/dc=mycompany,dc=com" />

    <property
        name   ="usernameFormat"
        value  ="uid={0},ou=people,dc=mycompany,dc=com" />

    <!-- Propietats opcionals -->

    <property
        name   ="userLookupNameFormat"
        value  ="uid={0},ou=people" />

    <property
        name   ="roleAttribute"
        value  ="title" />
    </bean>
```

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

En el cas que es requereixi configurar l'autentificació per mitjà de crides directes a SACE podem fer ús d'una classe diferent.

Per a realitzar les proves en desenvolupament podem instal·lar un VPN Client **per connectar-se al servidor SACE** (consultar el document 'Directori Corporatiu de la Generalitat de Catalunya - Guia d'integració d'aplicacions v3.7').

La configuració de l'accés al SACE es fa per mitjà d'un bean de la classe '`net.opentrends.openframe.services.security.acegi.SACEAuthenticationConfiguration'`, per a la que podrem definir les següents propietats:

Propietat	Requerit	Descripció
urlSACE	Sí	Url del servidor SACE. **Per exemple, la url del servidor de pre-producció és:** https://sace.**prepdc**.gencat.intranet/SACE/SACE_Logon.aspx?XMLIn= Segons l'entorn podem canviar el valor d'aquesta url: - dc, per producció - prepdc, per pre-producció - desenvdc, per desenvolupament
userNameFormat	Sí	Format del campo 'user name'. Els 2 valors possibles són:: - INTERNAL_CODE, codi intern - NIF
authoritiesbyUserNameQuery	No	Aquesta propietat permet especificar la query SQL per a recollir els rols dels usuaris. Per exemple, "SELECT {Nombre_usuario}, {rol} FROM {privilegios} WHERE {Nombre_usuario} =?

Exemple:

```
<bean    id="SACEAuthenticationConfiguration"
         class="net.opentrends.openframe.services.security.
                        SACEAuthenticationConfiguration">

   <property
   name   ="urlSACE"
   value  ="https://sace.prepdc.gencat.intranet/SACE/SACE_Logon.aspx?XMLIn="/>

   <property
```

```
            name   ="userNameFormat"
            value  ="NIF"/>
    </bean>
```

Configuració del Formulari d'Autentificació i la Seqüència d'Autentificació

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

Una vegada definides les fonts d'autentificació configurarem quin és el formulari d'autentificació i com ha de realitzar-se l'autentificació, potser en més d'una font. Per exemple, podriem considerar que en primer lloc s'autentifiqui en un esquema de base de dades i si no es troba en aquesta comprovar-ho en un LDAP corporatiu, etc.

Per a configurar el formulari i la seqüència es farà ús d'un bean de la classe 'net.opentrends.openframe.services.security.acegi.AuthenticationSecurityConfiguration' en un fitxer XML "application context" de Spring. La taula següent detalla les propietats:

Propietat	Requerit	Descripció
loginFormUrlValue	Sí	Url del formulari d'autentificació. El formulari ha de seguir unes normatives tal com es defineix en l'apartat 'Autentificació per Formulari' de la secció 'Utilització del Servei'
authenticationFailureUrlValue	Sí	Per defecte serà el mateix que 'loginFormUrlValue'. Si l'autentificació falla es presentarà la url indicada en el camp 'authenticationFailureUrValue'. Si és correcta, el navegador serà redirigit a la url inicial protegida que va ser sol·licitada per l'usuari (per a definir quins recursos són protegits consultar l'apartat 'Configuració de les urls per rol d'usuari')
filterProcessesUrl	Sí	Url del 'submit' en el formulari de login. tal com es defineix en l'apartat 'Autentificació per Formulari' de la secció 'Utilització del Servei'
authenticationProvidersConfigurationList	Sí	Llista de beans de configuració (BBDD o LDAP) que defineixen la seqüència de fonts de l'autentificació. S'intentarà l'autentificació de l'usuari / contrasenya en cadascun dels autentificadors definits en l'ordre tal i com apareixen al fitxer de a dalt a baix.

Exemple:

```
<bean id="SACEAuthenticationConfiguration1"
        ...
</bean>


<bean id="ldapAuthenticationConfiguration2"
        ...
</bean>

<bean id="databaseAuthenticationConfiguration3"
        ...
</bean>

<bean id="authenticationConfiguration"
class="net.opentrends.openframe.services.security.AuthenticationSecurityConfigurat
ion">

   <property      name="loginFormUrlValue"
                  value="/login.jsp" />

   <property      name="authenticationFailureUrlValue"
          value="/login.jsp" />

   <property      name="filterProcessesUrl"     value="/AppJava/acegiLogon" />

   <property      name="authenticationProvidersConfigurationList">
          <list>
                 <ref local="SACEAuthenticationConfiguration1" />
                 <ref local="ldapAuthenticationConfiguration2" />
                 <ref local="databaseAuthenticationConfiguration3" />
          </list>
   </property>
</bean>
```

2.3.5. Configuració de l'Autorització

Fitxer de configuració: openFrame-services-security.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

Podem definir l'autorització a 3 nivells:

- Configuració de les Autoritzacions d'urls per rol d'usuari
- Configuració de les Autoritzacions a nivell de classe i mètode
- Configuració de les Autoritzacions amb ACLs

Configuració de les Autoritzacions d'urls per rol d'usuari

L'exemple més normal i el que utilitzarem com a norma general en les nostres aplicacions, és donar accés segons els rols que té l'usuari. La configuració de les autoritzacions d'urls es fa per mitjà de la propietat "secureUrls" del bean d'openFrame 'authorizationConfiguration'.

Propietat	Requerit	Descripció

Propietat	Requerit	Descripció
secureUrls	Sí	Per a cada url que volem restringir, afegirem una línia amb el format següent: <Patró de URL> = <llista de rols separada per ‘,'> Exemples: /**/*view* = ROLE_GESTOR, ROLE_USUARIO /**/*delete* = ROLE_GESTOR /secure/** = ROLE_GESTOR En l'exemple, la primera línia significa que les urls que contenen la paraula “view” són autoritzades per als usuaris que tenen el rol GESTOR o el rol USUARI.

Exemple:

```
<bean    id="authorizationConfiguration"
         class="net.opentrends.openframe.services.security.
                                  AuthorizationSecurityConfiguration">

         <!-- Propietat opcional per les urls autoritzades -->
         <property    name   ="secureUrls">
         <value>
               /**/*read*   =     ROLE_GESTOR, ROLE_USUARIO
               /**/*delete* =     ROLE_GESTOR
               /secure/**   =     ROLE_GESTOR
               </value>
         </property>
   </bean>
```

Configuració de les Autoritzacions a nivell de classe i mètode

Per mitjà de proxies dinàmics lligats a aspectes (programació orientada a aspectes) podem configurar que determinats rols no tinguin accés a determinats mètodes de determinades classes.
La configuració de les autoritzacions a nivell de classe i mètode s'han de realitzar per mitjà de la propietat “secureBusinessObjects” del bean d'openFrame “authorizationConfiguration”. Per a aquest bean definirem les propietats següents:

Propietat	Requerit	Descripció

Propietat	Requerit	Descripció
secureBusinessObjects	Sí	Per a cada classe o mètode url que vulguem restringir, afegirem una línia amb el format següent: <Nom de classe full qualified o nom.mètode> = <llista rols separats per ‘,’> Exemple: com.business.UsersManager = ROLE_GESTOR com.business.CategoryManager.delete = ROLE_GESTOR com.business.CategoryManager.read = ROLE_GESTOR, X En l'exemple, la primera línia especifica que només els usuaris que tenen el rol GESTOR estan autoritzats a cridar els mètodes de la classe “UserManager”.

Exemple:

```
<bean    id="authorizationConfiguration"
         class="net.opentrends.openframe.services.security.
                                    AuthorizationSecurityConfiguration">

  <!-- Propietat opcional per les classes i mètodes autoritzats -->
  <property     name="secureBusinessObjects">
   <value>
   com.business.UsersManager   = ROLE_GESTOR
   com.business.CategoryManager.delete = ROLE_GESTOR
   com.business.CategoryManager.read = ROLE_GESTOR, ROLE_USUARIO
   </value>
  </property>

</bean>
```

Configuració de les Autoritzacions amb ACLS

Hi ha casos en que la protecció de les crides a mètodes no és suficient, necessitant protegir-se de diferent forma diferents instàncies d'una classe. En la majoria de casos, no fa falta cap configuració de fitxers XML.

La protecció a les instàncies necessita conéixer quin és l'identificador de la instància. Per defecte es considerarà el mètode 'getId()' de l'objecte. Si no existís aquest mètode podrem configurar quin és el mètode a utilitzar per a obtenir l'identificador de l'objecte.

Aquesta configuració es fa en el bean d'openFrame 'authorizationConfiguration' per mitjà de la propietat "domainObjectsIdGetters":

Propietat	Requerit	Descripció
domainObjectsIdGetters	Sí	Per a definir el mètode d'obtenció de l'identificador usarem la sintaxi següent: <Nom de la classe> = <Nom del métode> Exemple: com.business.ComplexDomainObject= getDomainObjectId()

Exemple:

```
<bean    id="authorizationConfiguration"
         class="net.opentrends.openframe.services.security.
                                    AuthorizationSecurityConfiguration">

  <!-- Propietat pels getters dels identificadors -->
  <property name      ="domainObjectsIdGetters">
    <value>
      com.business.ComplexDomainObject = getDomainObjectId
    </value>
  </property>

</bean>
```

2.3.6. Configuració del Servei d'Accés a Informació General

En casos concrets, des del codi voldrem obtenir informació relacionada amb l'usuari connectat. Per aquest objectiu s'ofereix una interfície 'SecurityService'. En l'actualitat s'ofereix una implementació basada en 'Acegi' que podrem utilitzar introduint la següetn configuració:

```
<bean id="securityService"
   class="net.opentrends.openframe.services.security.acegi.SecurityServiceAcegiImp
l"/>
```

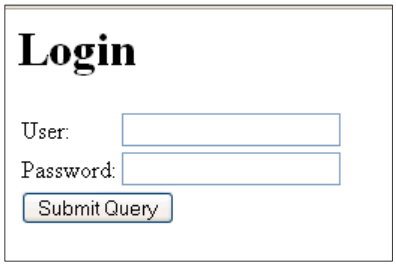
En aquest moment, segons la configuració d'injecció de beans (tal i com s'exposa al document de Servei de Configuració) podrem usar el servei des de qualsevol classe de l'aplicació.

2.4. Utilització del Servei

2.4.1. Autentificació per Formulari

Hi ha diverses possibilitats d'autentificació des de Web. En aquest apartat ens centrarem en l'autentificació amb formulari (per a més informació consultar la secció '1.10' del tutorial de 'Acegi').

És necessari crear una pàgina en què s'introdueixi l'usuari i el password.



Aquesta pàgina ha de complir els requisits següents:

1) El camp en que s'introdueix el nom d'usuari ha de tenir com a nom 'j_username'
2) El camp en que s'introdueixi el password ha de tenir com a nom 'j_password'
3) L'acció del formulari ha de ser la url especificada en el filtre en la propietat 'filterProcessesUrl' (per defecte '/j_acegi_security_check')

Podem adaptar el codi font següent en les nostres pàgines de Login. **El text en negreta i marcat** és codi reutilitzable:

```
<%@ taglib prefix='c' uri='http://java.sun.com/jstl/core'%>
<%@ page import="net.sf.acegisecurity.ui.AbstractProcessingFilter"%>
<%@ page
import="net.sf.acegisecurity.ui.webapp.AuthenticationProcessingFilter"%>
<%@ page import="net.sf.acegisecurity.AuthenticationException"%>

<html>
<head>
<title>Login</title>
</head>

<body>
<h1>Login</h1>

<p><%-- this form-login-page form is also used as the
        form-error-page to ask for a login again.
        --%>
<c:if test="${not empty param.login_error}">
  <font color="red"> Your login attempt was not successful, try
again.<BR>
  <BR>
  Reason:
<%=((AuthenticationException) session
  .getAttribute(AbstractProcessingFilter.ACEGI_SECURITY_LAST_EXCEPTION_K
EY))
.getMessage()%>
  </font>
</c:if>

<form action="<c:url value='j_acegi_security_check'/>" method="POST">
<table>
  <tr>
        <td>User:</td>
        <td>
                <input type='text' name='j_username'
                      <c:if test="${not empty param.login_error}">
value='<%=session.getAttribute(AuthenticationProcessingFilter.ACEGI_SECURITY_LAST
_USERNAME_KEY) %>'
                      </c:if>>
        </td>
  </tr>
  <tr>
        <td>Password:</td>
        <td>
                <input type='password' name='j_password'>
        </td>
  </tr>

  <tr>
        <td colspan='2'><input name="submit" type="submit"></td>
  </tr>
</table>

</form>

</body>
```

```
</html>
```

2.4.2. Utilització de l'Autorització en les nostres pàgines JSP

En cas de voler presentar determinat contingut depenent dels rols de l'usuari podem fer ús dels tags proporcionats per Acegi. Per a això, seguirem els passos següents:

1) Definició de la referència a la llibreria de tags

   Des de la versió JSP 1.2 no és necessari configurar en el fitxer 'web.xml' la referència a les llibreries. Podem referenciar directament per mitjà d'una url el jar, tal com es mostra a continuació:

```
<%@ taglib prefix="authz" uri="http://acegisecurity.sf.net/authz" %>
```

2) Utilitzar el tag "authorize" en la pàgina JSP

   Una vegada definida la referència a la llibreria i el seu prefix 'authz', podem incorporar el tag a la pàgina JSP per mitjà de '<authz:authorize>', en el que podrem definir els atributs següents:

Propietat	Requerit	Valor
ifAllGranted	No	Llista de rols separats per coma. L'usuari ha de tenir tots els rols perquè el contingut dins del tag (body) sigui mostrat.
ifAnyGranted:	No	Llista de rols separats per coma. L'usuari ha de tenir qualsevol dels rols perquè el contingut dins del tag (body) sigui mostrat.
ifNotGranted	No	Llista de rols separats per coma. L'usuari no ha de tenir cap dels rols perquè el contingut dins del tag (body) sigui mostrat.

Exemple:

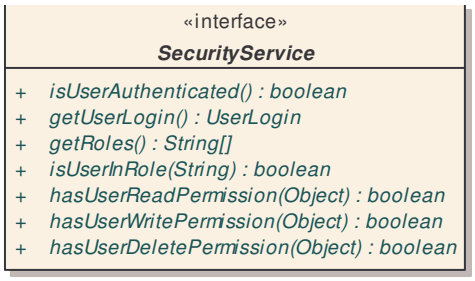
```
<authz:authorize ifAllGranted="ROLE_ADMIN">
  <td>
    <A HREF="del.htm?id="123">Borrar</A>
  </td>
```

```
</authz:authorize>
```

Per a més informació sobre el *tag* "authorize", consultar la pàgina
http://acegisecurity.sourceforge.net/docbook/acegi.html#N10C28

2.4.3. Obtenció d'Informació de Seguretat amb API

El servei de seguretat, definit amb el bean 'SecurityService' (tal i com s'explica en l'apartat de Configuració) conté una signatura d'operacions tal i com es mostra en el següent diagrama:



- `public boolean isUserAuthenticated()`

Retorna "true" si l'usuari està autentificat.

- `public UserLogin getUserLogin()`

Retorn la informació de l'usuari actual.

- `public Boolean isUserInRole(String role)`

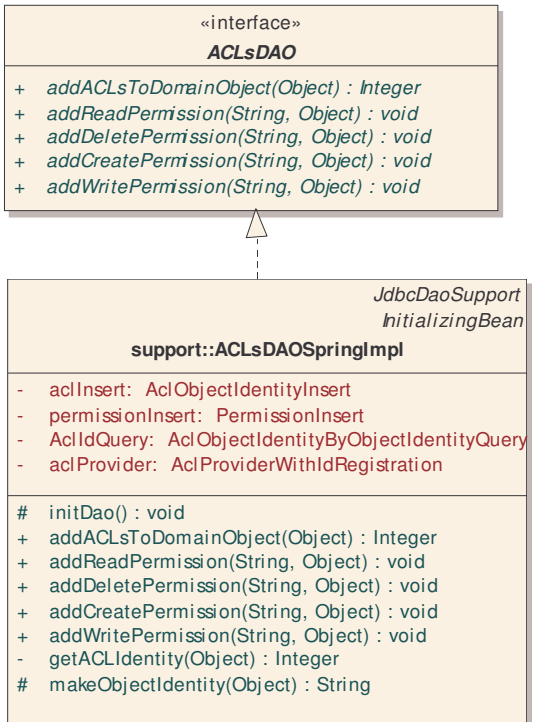
Comprova si l'usuari té el rol indicat com a paràmetre.

- `public Boolean hasUserXXXPermission(Object instancia), XXX = Read, Write, Delete`

Comprova si l'usuari té un permí determinat sobre un objecte.

2.4.4. Gestió dels ACLs mitjançant la API

Per a cada instància de la lògica de negoci podem afegir a la base de dades els permisos per mitjà de l'ús de la interfície 'ACLsDAO':



- `public void addReadPermission(String usuarioOrole, Object instancia)`

  Afegeix el permís *READ* (llegir) a un rol o un usuari "`usuarioOrole`" i una instància de la lògica de negoci "`instància`"

Altres mètodes públics de la classe tenen un funcionament semblant (Delete, CREATE, Write). Per a més referència consultar el javadoc de la interfície 'ACLsDAO´.

2.5. Eines de Suport

2.5.1. Servidor LDAP de proves: openLDAP

Els diferents passos per a instal·lar openLDAP i importar un directori LDAP d'exemple són:

- Baixar openLDAP per a Windows http://lucas.bergmans.us/hacks/openldap/download i instal·lar-ho (versió 2.2.19).

- Canviar la configuració per defecte de openldap. Copiar les dades següents en un fitxer slapd.conf. Copiarem el slapd.conf en la mateixa carpeta que openLDAP.

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
ucdata-path     ./ucdata
include         ./etc/schema/core.schema
include         ./etc/schema/cosine.schema
include         ./etc/schema/inetorgperson.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org

pidfile         ./var/run/slapd.pid
argsfile ./var/run/slapd.args

# BDB database definitions
#######################################################################

###database     bdb
###suffix              "dc=my-domain,dc=com"
###rootdn              "cn=Manager,dc=my-domain,dc=com"


database bdb
suffix dc="mycompany",dc="com"
rootdn "cn=Manager,dc=mycompany,dc=com"

rootpw secret

directory       ./var/openldap-data
index    objectClass  eq
```

- Obrir una pantalla "DOS command", anar a la carpeta on hem instal·lat el programa i arrancar openLDAP amb la comanda

```
.\slapd -d 1
```

Si tot ha funcionat bé, hauríem de veure una sortida com la següent:

- Copiar les dades següents en un fitxer setup.ldif. Aquest fitxer conté un directori LDAP de l'empresa "mycompany.com" amb 2 persones: "gestoruser" i "usuari". Copiarem el setup.ldif en la mateixa carpeta que openLDAP.

```
### Top level definition
#dn: dc=mycompany,dc=com
#objectClass: top
#objectClass: dcObject
#objectClass: domain
#dc: mycompany

### organizationalUnit : PEOPLE
# Definition of people
dn: ou=people,dc=mycompany,dc=com
objectClass: top
objectClass: organizationalUnit
ou: people

# Gestor User
dn: uid=gestoruser,ou=people,dc=mycompany,dc=com
objectClass: person
objectClass: inetOrgPerson
cn: State App
displayName: App Admin
givenName: App
mail: gestor@fake.org
title: ROLE_ADMIN
sn: Gestor
uid: gestoruser
userPassword: gestorpassword

# usuario normal
dn: uid=usuario,ou=people,dc=mycompany,dc=com
objectClass: person
objectClass: inetOrgPerson
cn: State App
displayName: App Admin
givenName: App
mail: usuario@fake.org
title: ROLE_USER
sn: Usuario
uid: usuario
userPassword: usuariopassword
```

- Obrir una altra pantalla "DOS command", anar a la carpeta on hem instal·lat el programa i importar les dades amb la comanda:

```
ldapadd -x -D "cn=Manager,dc=mycompany,dc=com" -W -f setup.ldif
```

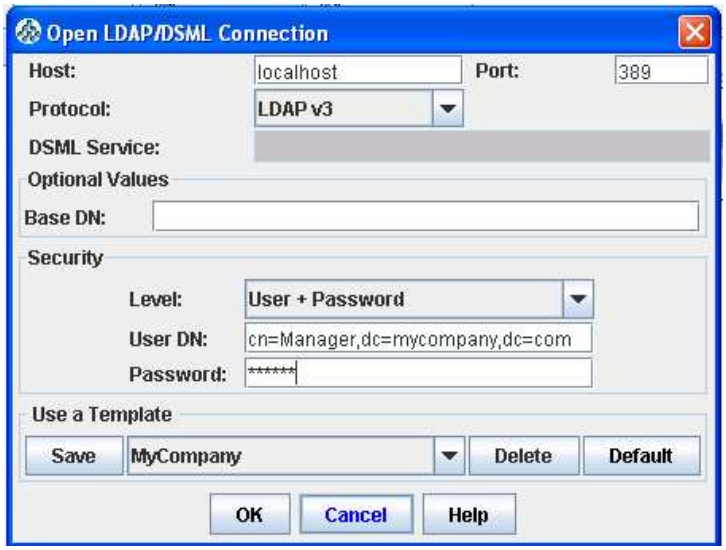
La contrasenya per defecte és "secret".

2.5.2. Jxplorer

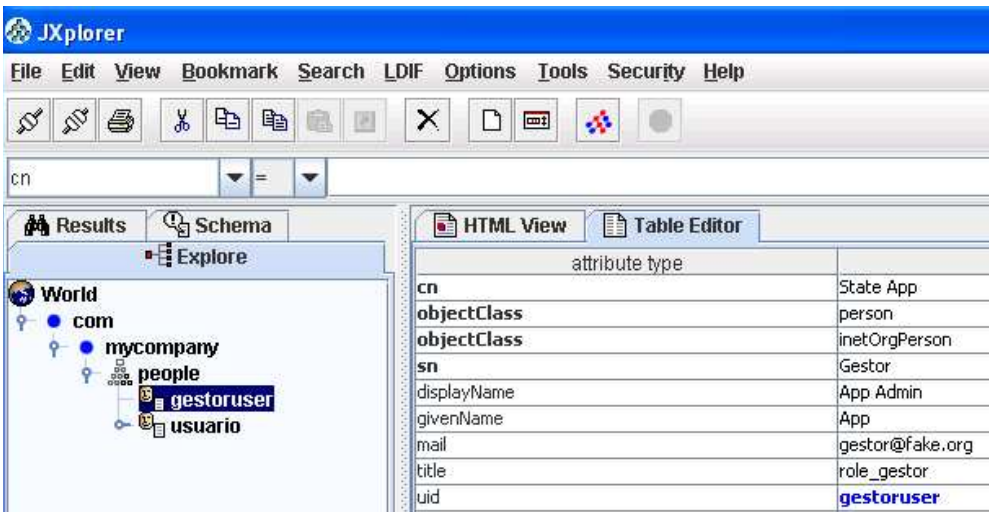
Comprovarem que la importació de dades ha funcionat amb Jxplorer, un client LDAP Java i opensource.

- Baixar Jxplorer a la url http://sourceforge.net/projects/jxplorer/ i instal·lar-ho
- Prémer el botó per a connectar-se al nostre directori LDAP.

La contrasenya per defecte és "secret". La pantalla següent mostra els valors de la diferents paràmetres:



- Si tot ha funcionat bé, hauríem de veure la pantalla següent:



3. Exemples

3.1. Exemple de configuració d'autentificació amb BBDD, SACE i LDAP

En aquesta part es descriuen 2 dels exemples més comuns d'autentificació:

- Autentificació per SACE
- Autentificació per BBDD ja existent

3.1.1. Exemple de configuració amb autentificació SACE

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



<bean id="SACEAuthenticationConfiguration"
   class="net.opentrends.openframe.services.security.SACEAuthenticationConfigurati
on">

   <property name="urlSACE"
value="https://sace.prepdc.gencat.intranet/SACE/SACE_Logon.aspx?XMLIn=" />
   <property name="userNameFormat" value="NIF" />
</bean>


<bean id="authenticationConfiguration"
class="net.opentrends.openframe.services.security.AuthenticationSecurityConfigurat
ion">

   <property name="loginFormUrlValue" value="/login.jsp" />

   <property      name="authenticationFailureUrlValue"
          value="/login.jsp" />


   <property name="filterProcessesUrl" value="/AppJava/acegiLogin" />

   <property name="authenticationProvidersConfigurationList">
   <list>
         <ref local="SACEAuthenticationConfiguration" />
   </list>
   </property>
</bean>


   <bean id="authorizationConfiguration"
```

```
class="net.opentrends.openframe.services.security.AuthorizationSecurityConfigurati
on">
          <property name="rolesList">
                <list>
                      <value>ROLE_ADMIN</value>
                      <value>ROLE_USER</value>
                </list>
          </property>

          <property name="secureUrls">
                <value>
                /**/*.do* =    ROLE_USER,ROLE_ADMIN
                </value>
          </property>

   </bean>

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

</beans>
```

Els beans openFrame que usen "Acegi" estan continguts en el jar del servei de seguretat, per la qual cosa haurem d'importar aquest recurs en el nostre "application context" de Spring amb la línia següent:

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

3.1.2. Autentificació per BBDD amb una BBDD ja existent

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


   <bean id="existingDatabaseAuthenticationConfiguration"

   class="net.opentrends.openframe.services.security.DatabaseAuthenticationConfigu
ration">

          <property name="passwordEncoderClass"

   value="net.sf.acegisecurity.providers.encoding.PlaintextPasswordEncoder" />
```

```
          <property name="usersByUserNameQuery" value="SELECT username,password
FROM users WHERE username = ?" />
          <property name="authoritiesbyUserNameQuery" value="SELECT username, role
FROM user_roles WHERE username =?" />

   </bean>


<bean id="authenticationConfiguration"
class="net.opentrends.openframe.services.security.AuthenticationSecurityConfigurat
ion">

   <property name="loginFormUrlValue" value="/login.jsp" />

   <property      name="authenticationFailureUrlValue"
          value="/login.jsp" />


   <property name="filterProcessesUrl" value="/AppJava/acegiLogin" />

   <property name="authenticationProvidersConfigurationList">
   <list>
         <ref local=" existingDatabaseAuthenticationConfiguration" />
   </list>
   </property>
</bean>


   <bean id="authorizationConfiguration"
class="net.opentrends.openframe.services.security.AuthorizationSecurityConfigurati
on">
          <property name="rolesList">
                 <list>
                        <value>ROLE_ADMIN</value>
                        <value>ROLE_USER</value>
                 </list>
          </property>

          <property name="secureUrls">
                 <value>
                 /**/*.do* =    ROLE_USER,ROLE_ADMIN
                 </value>
          </property>

   </bean>

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

</beans>
```

3.2. Exemple de formulari de login amb JSTL i Struts

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
```

```
<%@ page import="net.sf.acegisecurity.ui.AbstractProcessingFilter"%>
<%@ page import="net.sf.acegisecurity.ui.webapp.AuthenticationProcessingFilter"%>
<%@ page import="net.sf.acegisecurity.AuthenticationException"%>

<style type="text/css">

<%-- this form-login-page form is also used as the
         form-error-page to ask for a login again.
         --%>
<c:if test="${not empty param.login_error}">
    <bean:message key="jsp.login.error" />
</c:if>

<bean:message key="jsp.login.message" />

<form action="<c:url value='/AppJava/acegiLogin'/>" method="post">

   <table width="100%" border="0">
      <tr>
        <td><bean:message key="jsp.login.user" />:</td>
        <td><input type='text' name='j_username'
                  <c:if test="${not empty param_error}">value='<%=
session.getAttribute(AuthenticationProcessingFilter.ACEGI_SECURITY_LAST_USERNAME_K
EY) %>'</c:if>></td>
      </tr>
      <tr>
        <td><bean:message key="jsp.login.password" />:</td>
        <td><input type='password' name='j_password'></td>
      </tr>
      <tr>
        <td colspan="2" align="right"><input type="submit" value="<bean:message
key="jsp.includes.submit"/>          " /></td>
      </tr>
    </table>

</form>
```

3.3. Exemple de configuració d'ACLs

Volem donar els permisos "llegir" i "escriure" (READ +WRITE) a un usuari "usuari1" per a la instància de la lògica de negoci "Categoria Papagais".

Primer, recollim el DAO de seguretat d'openFrame:

```
// Get the DAO from the Spring context
ACLsDAO securityDAO = (ACLsDAO) applicationContext.getBean("securityDao");
```

Segon, recollim la instància de la lògica de negoci "Categoria Papagais" , per mitjà d'un DAO:

```
// Get the business object using a DAO
Category papagayo = CategoryDAO.read("papagayo");
```

Finalment podem afegir els permisos per a aquesta instància per l'usuari "usuari1":

```
// Add permissions for user "usuario1" for the papagayo
securityDAO.addReadPermission("usuario1", papagayo);
securityDAO.addWritePermission("usuario1", papagayo);
```