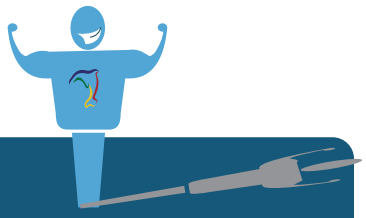
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

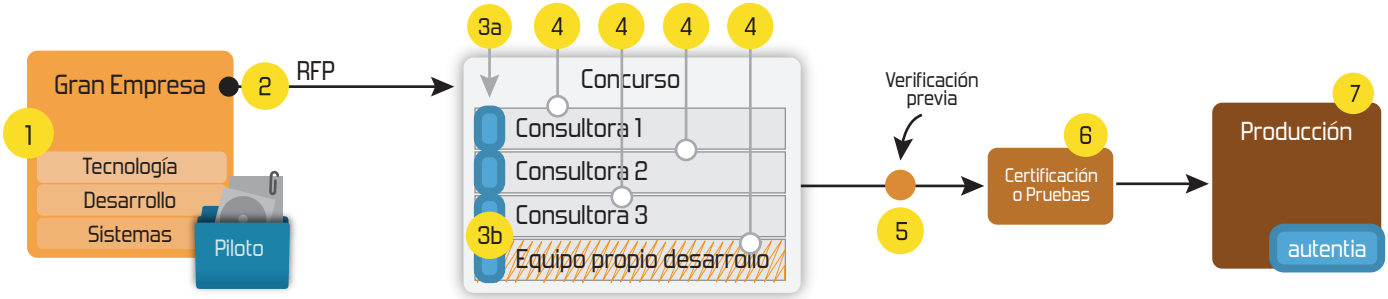
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Jose Manuel Sánchez Suárez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2013-03-04

Spring Security: haciendo uso de un servidor LDAP embebido.

0. Índice de contenidos.

1. Introducción

Ya hemos tenido oportunidad en adictos de examinar las posibilidades de Spring Security a la hora de gestionar la autenticación y autorización en nuestras aplicaciones; más allá de ser un simple filtro. De hecho hacemos uso intensivo del mismo y forma parte de nuestra arquitectura de referencia.

En este tutorial vamos a ver cómo puede darnos soporte para configurar un servidor LDAP embebido con el objetivo de incluirlo o bien dentro de nuestros tests de integración o para dar soporte a nuestras aplicaciones web sin necesidad de tener un servidor ldap real. Así lo podríamos "levantar" para dar soporte a la autenticación SSO de CAS sin necesidad de disponer de un servidor físico.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.4 GHz Intel Core i7, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Lion 10.7.4
- Spring 3.2.1.RELEASE
- Spring Security 3.1.3.RELEASE

3. Configuración.

Como de costumbre, haciendo uso de maven, lo primero es declarar nuestras dependencias en el fichero pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maver
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.autentia.tutoriales.bpm.ldap</groupId>
    <artifactId>spring-security-ldap-embedded</artifactId>
    <packaging>jar</packaging>
    <version>1.0.0-SNAPSHOT</version>
    <name>spring-security-ldap-embedded</name>

    <properties>
        <java.version>1.6</java.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <spring.version>3.2.1.RELEASE</spring.version>
        <spring.security.version>3.1.3.RELEASE</spring.security.version>
        <slf4j.version>1.6.4</slf4j.version>
        <log4j.version>1.2.16</log4j.version>
```

```
    </properties>

    <repositories>
        <repository>
            <id>com.springsource.repository.bundles.release</id>
            <name>EBR Spring Release Repository</name>
            <url>http://repository.springsource.com/maven/bundles/release</url>
        </repository>
    </repositories>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.5.1</version>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <!-- spring -->

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- spring security -->
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-core</artifactId>
            <version>${spring.security.version}</version>
                <exclusions>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.aspectj</groupId>
                        <artifactId>aspectjweaver</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-tx</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.aspectj</groupId>
                        <artifactId>aspectjrt</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-core</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-aop</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-expression</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-beans</artifactId>
                    </exclusion>
                </exclusions>
        </dependency>

        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>${spring.security.version}</version>
                <exclusions>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.aspectj</groupId>
                        <artifactId>aspectjweaver</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-tx</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.aspectj</groupId>
                        <artifactId>aspectjrt</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
```

» Gestión de expedientes en el ámbito de las Administraciones Públicas (III): BPM y la gestión de procesos de negocio.

» jBPM5: usando nuestra propia base de datos.

» AngularJS: primeros pasos.

Últimos Tutoriales del Autor

» Introducción a Guvnor

» Gestión de expedientes en el ámbito de las Administraciones Públicas (IV): buscando en las forjas una solución.

» Gestión de expedientes en el ámbito de las Administraciones Públicas (III): BPM y la gestión de procesos de negocio.

» jBPM5: usando nuestra propia base de datos.

» Gestión de expedientes en el ámbito de las Administraciones Públicas (II): requisitos.

Últimas ofertas de empleo

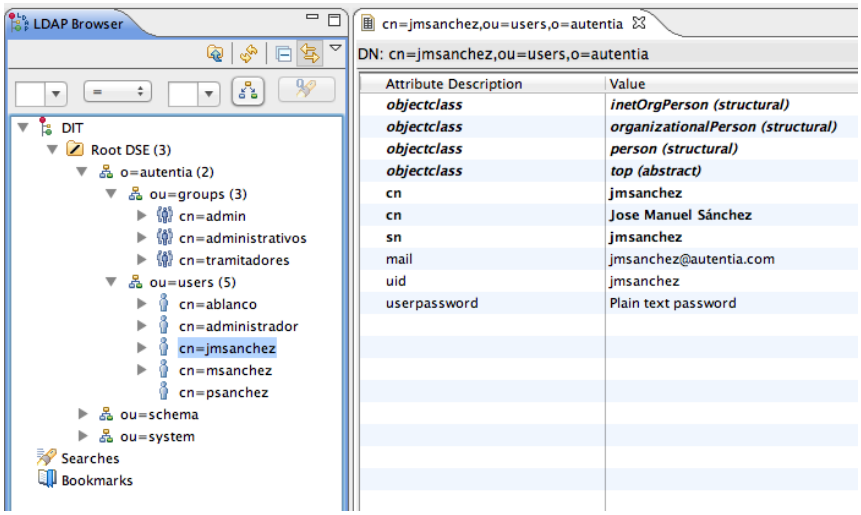
```
                        <artifactId>spring-core</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-aop</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-expression</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-beans</artifactId>
                    </exclusion>
                </exclusions>
        </dependency>

        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-ldap</artifactId>
            <version>${spring.security.version}</version>
                <exclusions>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-tx</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-core</artifactId>
                    </exclusion>
                    <exclusion>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-beans</artifactId>
                    </exclusion>
                </exclusions>
        </dependency>
        <dependency>
            <groupId>org.apache.directory.server</groupId>
            <artifactId>apacheds-all</artifactId>
            <version>1.5.5</version>
        </dependency>

        <!-- logging -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>${log4j.version}</version>
        </dependency>

        <!-- Test -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
            <scope>test</scope>
        </dependency>

    </dependencies>

</project>
```

El servidor que usaremos será Apache Directory que se alimentará en su arranque de un fichero ldif (autentia-identity-repository.ldif), con la siguiente información sobre nuestra organización:

```
version: 1

dn: o=autentia
objectClass: organization
objectClass: extensibleObject
objectClass: top
o: autentia

dn: ou=users,o=autentia
objectClass: extensibleObject
objectClass: organizationalUnit
objectClass: top
ou: users

dn: ou=groups,o=autentia
objectClass: extensibleObject
objectClass: organizationalUnit
objectClass: top
```

```
ou: groups

dn: cn=administrativos,ou=groups,o=autentia
objectClass: groupOfUniqueNames
objectClass: top
objectClass: group
cn: administrativos
uniqueMember: cn=jmsanchez,ou=users,o=autentia
uniqueMember: cn=psanchez,ou=users,o=autentia

dn: cn=tramitadores,ou=groups,o=autentia
objectClass: groupOfUniqueNames
objectClass: top
objectClass: group
cn: tramitadores
uniqueMember: cn=ablanco,ou=users,o=autentia
uniqueMember: cn=msanchez,ou=users,o=autentia

dn: cn=admin,ou=groups,o=autentia
objectClass: groupOfUniqueNames
objectClass: top
objectClass: group
cn: admin
uniqueMember: cn=administrador,ou=users,o=autentia

dn: cn=jmsanchez,ou=users,o=autentia
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
cn: Jose Manuel Sánchez
sn: jmsanchez
uid: jmsanchez
mail: jmsanchez@autentia.com
userPassword:: cGFzcw==

dn: cn=psanchez,ou=users,o=autentia
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
cn: Pablo Sánchez
sn: psanchez
uid: psanchez
mail: psanchez@autentia.com
userPassword:: cGFzcw==

dn: cn=msanchez,ou=users,o=autentia
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
cn: Mario Sánchez
sn: msanchez
uid: msanchez
mail: msanchez@autentia.com
userPassword:: cGFzcw==

dn: cn=ablanco,ou=users,o=autentia
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
cn: Alfonso Blanco
sn: ablanco
uid: ablanco
mail: ablanco@autentia.com
userPassword:: cGFzcw==

dn: cn=administrador,ou=users,o=autentia
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
cn: admin
sn: admin
uid: administrador
userPassword:: cGFzcw==
```

Tenemos una pequeña jerarquía, con dos grupos de usuarios: administrativos y tramitadores (además del grupo de administradores) y varios usuarios asociados a los mismos que montará una estructura como la siguiente:



Nota: para obtener esta captura he puesto un punto de ruptura en el test que veremos en el siguiente punto y he comprobado el acceso al directorio, por el puerto configurado, con la herramienta Apache Directory Studio

La configuración del servidor LDAP la incluimos en un fichero de configuración Spring, con un contenido similar al siguiente (src/main/resources/spring-config/ldapServer.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:security="http://www.springframework.org/schema/security"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/security
            http://www.springframework.org/schema/security/spring-security.xsd">

    <security:ldap-server ldif="classpath:autentia-identity-repository.ldif" root="o=aut

</beans>
```

En el propio espacio de nombres del contexto de seguridad disponemos de una configuación para ldap donde configuramos el fuente ldif con el contenido de nuestra organización, el nodo raíz para acceder al mismo y el puerto, a través del cuál podemos acceder al mismo.

4. Test.

Vamos a escribir, con el soporte de Spring, un test que compruebe el acceso y recuperación de usuarios y grupos del directorio:

```
package com.autentia.tutoriales.spring.security.ldap;

import static org.junit.Assert.assertEquals;

import java.util.List;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;


@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration({"classpath:spring-config/ldapServer.xml","classpath:spring-confic
public class PersonRepositoryIntegrationTest {

    @Autowired
    private PersonRepository personRepository;

    @Test
    public void shouldFindUserById(){
        final String userId = "jmsanchez";
        final List<Person> persons = personRepository.getByUID(userId);
        assertEquals(1, persons.size());
        assertEquals("Jose Manuel Sánchez", persons.get(0).getName());
    }

    @Test
    public void shouldFindUsersByGroupId(){
        final String groupId = "administrativos";
        final List<Person> persons = personRepository.getByGroupUID(groupId);
        assertEquals(2, persons.size());
    }

}
```

En este punto nuestro test estará en ROJO (errores de compilación), porque:

- no disponemos de la clase PersonRepository, que nos proporcionará acceso al repositorio de usuarios o personas,
- no tenemos la clase Person, que tendrá las propiedades de un usuario, y
- hemos separado la configuración del servidor de la configuración del contexto para acceder al mismo (aún no hemos

visto el contenido del fichero de configuración ldapContext.xml).

Veamos la interfaz de servicio que hemos pensado para el repositorio de personas

```
package com.autentia.tutoriales.spring.security.ldap;

import java.util.List;

public interface PersonRepository {

    List<Person> getByUID(String userId);

    List<Person> getByGroupUID(String groupId);

}
```

Y ahora el contenido del POJO que tendrá las propiedades de un usuario:

```
package com.autentia.tutoriales.spring.security.ldap;

public class Person {

    private String uid;

    private String name;

    private String email;

    public String getUid() {
        return uid;
    }

    public void setUid(String uid) {
        this.uid = uid;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
```

Ahora no tendremos errores de compilación, pero fallará la ejecución del test:

5. Implementación.

Veamos el contenido que proponemos para el repositorio de usuarios:

```
package com.autentia.tutoriales.spring.security.ldap;

import java.util.ArrayList;
import java.util.List;

import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.naming.directory.Attributes;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ldap.core.AttributesMapper;
import org.springframework.ldap.core.LdapTemplate;

public class PersonLdapRepository implements PersonRepository{

    private LdapTemplate ldapTemplate;

    @Autowired
    public PersonLdapRepository(LdapTemplate ldapTemplate){
        this.ldapTemplate = ldapTemplate;
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<Person> getByUID(String userId) {
        return ldapTemplate.search("", "(uid=" + userId + ")", personMapper());
    }

    @Override
    public List<Person> getByGroupUID(String groupId) {
        final List<Person> persons = new ArrayList<Person>();
        @SuppressWarnings("unchecked")
        final List<String> userIds = (List<String>) ldapTemplate.search("", "(&(objectCl
        for (String userId : userIds) {
            persons.add(getByUID(userId).get(0));
        }
        return persons;
    }

    private AttributesMapper personMapper() {
```

```
        return new AttributesMapper() {

            @Override
            public Person mapFromAttributes(final Attributes attributes) throws NamingE
                final Person person = new Person();
                person.setUid( (String) attributes.get("uid").get());
                person.setName( (String) attributes.get("cn").get());
                person.setEmail( (String) attributes.get("mail").get());
                return person;
            }
        };
    }

    private AttributesMapper groupMapper() {
        return new AttributesMapper() {

            @Override
            public List<String> mapFromAttributes(final Attributes attributes) throws Na
                final List<String> userIds = new ArrayList<String>();
                final NamingEnumeration<?> uniquemembers = attributes.get("uniquemember"
                while (uniquemembers.hasMore()) {
                    String userId = uniquemembers.next().toString().split(",")[0].split(
                    userIds.add(userId);
                }
                return userIds;
            }
        };
    }

}
```

Básicamente, se apoya en una clase de plantilla de Spring para realizar las consultas al servidor ldap y se encarga de mapear el resultado de las consultas en un objeto de tipo Person (lo que haría un típico Dao).

La configuración que nos resta es la del fichero (src/main/resources/spring-config/ldapContext.xml) que va a configurar el contexto de acceso al servidor ldap que tenemos levantado para el entorno de tests, lo va a asociar a la plantilla de spring y ésta será inyectada en nuestro servicio.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:security="http://www.springframework.org/schema/security"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/security
            http://www.springframework.org/schema/security/spring-security.xsd">

    <bean id="contextSource" class="org.springframework.ldap.core.support.LdapContextSou
        init-method="afterPropertiesSet">
        <property name="url" value="ldap://localhost:9898" />
        <property name="base" value="o=autentia" />
        <property name="userDn" value="uid=admin,ou=system" />
        <property name="password" value="secret" />
    </bean>

    <bean id="ldapTemplate" class="org.springframework.ldap.core.LdapTemplate"
        init-method="afterPropertiesSet">
        <constructor-arg ref="contextSource"/>
    </bean>

    <bean id="personRepository" class="com.autentia.tutoriales.spring.security.ldap.Pers
        <constructor-arg ref="ldapTemplate"/>
    </bean>

</beans>
```

6. Referencias.

- http://static.springsource.org/spring-security/site/docs/3.0.x/reference/ldap.html
- http://static.springsource.org/spring-ldap/docs/1.3.x/reference/pdf/spring-ldap-reference.pdf
- http://krams915.blogspot.com.es/2011/01/spring-security-mvc-using-embedded-ldap.html

7. Conclusiones.

Como siempre, con el soporte de Spring, lo difícil se termina convirtiendo en algo fácil y vamos allanando el terreno para configurar un sistema de identidad único en nuestro entorno.

Un saludo.

Jose

jmsanchez@autentia.com