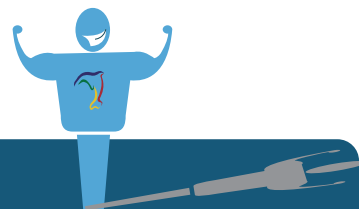


¿Qué ofrece Autentia Real Business Solutions S.L?

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

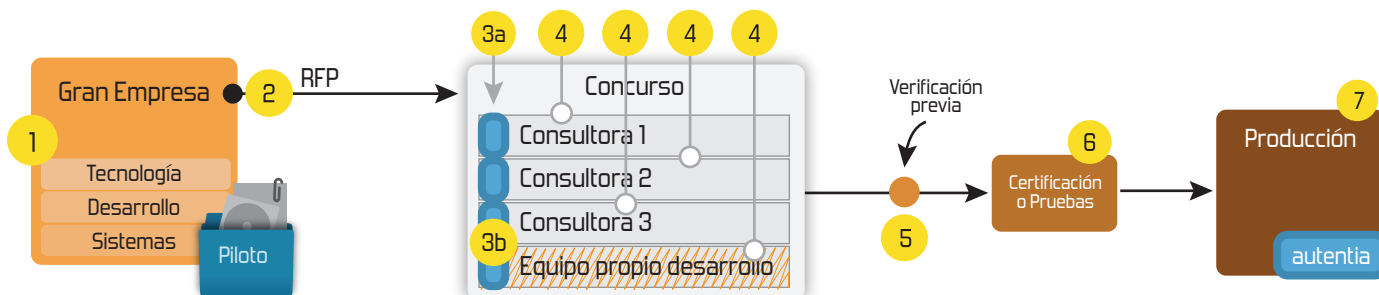
1. Desarrollo de componentes y proyectos a medida



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

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

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

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



Entra en Adictos a través de

E-mail

Contraseña

Entrar [Deseo registrarme](#) [Olvidé mi contraseña](#)

[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestros libros](#) [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) ApacheDS: tests de integración contra un servidor LDAP embebido.



Jose Manuel Sánchez Suárez

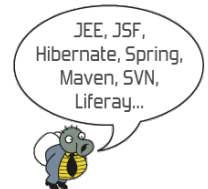
Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

[Ver todos los tutoriales del autor](#)

Catálogo de servicios Autentia



Fecha de publicación del tutorial: 2013-12-26

Tutorial visitado 48 veces [Descargar en PDF](#)

ApacheDS: tests de integración contra un servidor LDAP embebido.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Configuración.
- 4. Test de integración.
- 5. Implementación.
- 6. Referencias.
- 7. Conclusiones.

1. Introducción

Vimos hace poco cómo disponer de un [servidor LDAP embebido](#) en nuestro entorno de tests de integración con el soporte de Spring Security, dentro del contexto de inyección de dependencias de Spring.

En este tutorial vamos a ver cómo hacer lo mismo pero sin disponer del soporte de Spring y, como consecuencia, tampoco de Spring Security.

Será tan similar que el servidor LDAP que usaremos será también Apache Directory Server.

2. Entorno.

El tutorial, y el código que contiene, han sido escritos usando el siguiente entorno:

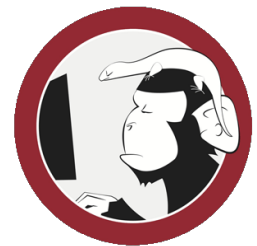
- Hardware: Portátil MacBook Pro 15' (2.4 GHz Intel Core i7, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Lion 10.7.4

3. Configuración.

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

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>com.autentia.tutorial.apacheDS</groupId>
4   <artifactId>apacheDS-integration-tests</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6
7   <properties>
8     <java.version>1.7</java.version>
9     <apache.ds.version>2.0.0-M15</apache.ds.version>
10  </properties>
11
12  <build>
13    <plugins>
14      <plugin>
15        <groupId>org.apache.maven.plugins</groupId>
16        <artifactId>maven-compiler-plugin</artifactId>
17        <version>2.5.1</version>
18        <configuration>
19          <source>${java.version}</source>
20          <target>${java.version}</target>
21        </configuration>
22      </plugin>
23    </plugins>
  
```



Síguenos a través de:



Últimas Noticias

» [IX Autentia Cycling Day \(ACTUALIZADO\)](#)

» Autentia en la carrera de las empresas

» Spring 4.0 ¿qué hay de nuevo amigo?

» [Torneo de pádel solidario AMEB](#)

» Próxima charla: Gradle como alternativa a Maven para la construcción de proyectos en Java

[Histórico de noticias](#)

Últimos Tutoriales

» ¿Mockear métodos estáticos?, con el soporte de PowerMock.

» Haciendo un cliente de Twitter en Android.

» ¿Endemoniado por lo lento que es Gradle en el arranque? Aprende a controlar su Daemon, y vuela!

» Manipulación de datos en MongoDB mediante Aggregation Pipeline.

```

24 </build>
25
26 <dependencies>
27 <!-- Apache DS dependencies -->
28 <dependency>
29 <groupId>org.apache.directory.server</groupId>
30 <artifactId>apacheds-all</artifactId>
31 <version>${apache.ds.version}</version>
32 <exclusions>
33 <exclusion>
34 <groupId>org.apache.directory.shared</groupId>
35 <artifactId>shared-ldap-schema</artifactId>
36 </exclusion>
37 <exclusion>
38 <groupId>org.apache.directory.api</groupId>
39 <artifactId>api-ldap-schema-data</artifactId>
40 </exclusion>
41 </exclusions>
42 </dependency>
43
44 <dependency>
45 <groupId>org.apache.directory.server</groupId>
46 <artifactId>apacheds-server-integ</artifactId>
47 <version>${apache.ds.version}</version>
48 <exclusions>
49 <exclusion>
50 <groupId>org.apache.directory.shared</groupId>
51 <artifactId>shared-ldap-schema</artifactId>
52 </exclusion>
53 <exclusion>
54 <groupId>org.apache.directory.api</groupId>
55 <artifactId>api-ldap-schema-data</artifactId>
56 </exclusion>
57 <exclusion>
58 <groupId>org.apache.directory.jdbm</groupId>
59 <artifactId>apacheds-jdbm</artifactId>
60 </exclusion>
61 </exclusions>
62 </dependency>
63
64 <dependency>
65 <groupId>org.slf4j</groupId>
66 <artifactId>slf4j-api</artifactId>
67 <version>1.6.0</version>
68 </dependency>
69 <dependency>
70 <groupId>org.slf4j</groupId>
71 <artifactId>slf4j-log4j12</artifactId>
72 <version>1.6.0</version>
73 </dependency>
74 <dependency>
75 <groupId>log4j</groupId>
76 <artifactId>log4j</artifactId>
77 <version>1.2.17</version>
78 </dependency>
79
80 <dependency>
81 <groupId>junit</groupId>
82 <artifactId>junit</artifactId>
83 <version>4.11</version>
84 <scope>test</scope>
85 </dependency>
86
87 </dependencies>
88 </project>

```

» Hello World en iOS sin
Storyboard

Últimos Tutoriales del Autor

» ¿Mockear métodos
estáticos?, con el soporte de
PowerMock.

» Integración de la gestión
de proyecto de Redmine en
Eclipse con el soporte de
Mylyn.

» Omnifaces: una librería de
utilidades para JSF2

» JUnit test runners

» Ejecución de un análisis en
sonar con el soporte de una
tarea ant.

Últimas ofertas de empleo

2011-09-08

 Comercial - Ventas -
MADRID.

2011-09-03

 Comercial - Ventas -
VALENCIA.

2011-08-19

 Comercial - Compras -
ALICANTE.

2011-07-12

 Otras Sin catalogar -
MADRID.

2011-07-06

 Otras Sin catalogar -
LUGO.

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:

```

1 version: 1
2
3 dn: O=autentia
4 changetype: add
5 objectClass: extensibleObject
6 objectClass: organization
7 objectClass: top
8 description: autentia
9
10 dn: ou=users,o=autentia
11 changetype: add
12 objectClass: extensibleObject
13 objectClass: organizationalUnit
14 objectClass: top
15 ou: users
16
17 dn: ou=groups,o=autentia
18 changetype: add
19 objectClass: extensibleObject
20 objectClass: organizationalUnit
21 objectClass: top
22 ou: groups
23
24 dn: cn=administrativos,ou=groups,o=autentia
25 changetype: add
26 objectClass: groupOfUniqueNames
27 objectClass: top
28 cn: administrativos
29 uniqueMember: cn=jmsanchez,ou=users,o=autentia
30 uniqueMember: cn=psanchez,ou=users,o=autentia
31
32 dn: cn=tramitadores,ou=groups,o=autentia
33 changetype: add
34 objectClass: groupOfUniqueNames
35 objectClass: top
36 cn: tramitadores
37 uniqueMember: cn=ablanc,ou=users,o=autentia
38 uniqueMember: cn=msanchez,ou=users,o=autentia
39
40 dn: cn=admin,ou=groups,o=autentia

```

```

41 changetype: add
42 objectClass: groupOfUniqueNames
43 objectClass: top
44 cn: admin
45 uniqueMember: cn=administrador,ou=users,o=autentia
46
47 dn: cn=jmsanchez,ou=users,o=autentia
48 changetype: add
49 objectClass: organizationalPerson
50 objectClass: person
51 objectClass: inetOrgPerson
52 objectClass: top
53 cn: Jose Manuel Sánchez
54 sn: jmsanchez
55 uid: jmsanchez
56 mail: jmsanchez@autentia.com
57 userPassword:: cGFzcw==
58
59 dn: cn=psanchez,ou=users,o=autentia
60 changetype: add
61 objectClass: organizationalPerson
62 objectClass: person
63 objectClass: inetOrgPerson
64 objectClass: top
65 cn: Pablo Sánchez
66 sn: psanchez
67 uid: psanchez
68 mail: psanchez@autentia.com
69 userPassword:: cGFzcw==
70
71 dn: cn=msanchez,ou=users,o=autentia
72 changetype: add
73 objectClass: organizationalPerson
74 objectClass: person
75 objectClass: inetOrgPerson
76 objectClass: top
77 cn: Mario Sánchez
78 sn: msanchez
79 uid: msanchez
80 mail: msanchez@autentia.com
81 userPassword:: cGFzcw==
82
83 dn: cn=ablanco,ou=users,o=autentia
84 changetype: add
85 objectClass: organizationalPerson
86 objectClass: person
87 objectClass: inetOrgPerson
88 objectClass: top
89 cn: Alfonso Blanco
90 sn: ablanco
91 uid: ablanco
92 mail: ablanco@autentia.com
93 userPassword:: cGFzcw==
94
95 dn: cn=administrador,ou=users,o=autentia
96 changetype: add
97 objectClass: organizationalPerson
98 objectClass: person
99 objectClass: inetOrgPerson
100 objectClass: top
101 cn: admin
102 sn: admin
103 uid: administrador
104 userPassword:: cGFzcw==

```

Al igual que en el tutorial anterior, tenemos una pequeña jerarquía, con dos grupos de usuarios: administrativos y tramitadores (además del grupo de administradores) y varios usuarios asociados a los mismos.

4. Test de integración.

Vamos a escribir, un test que comprueba el acceso y recuperación de un usuario:

```

1 package com.autentia.tutorial.apacheDS.accounts;
2
3 import static org.junit.Assert.assertEquals;
4 import static org.junit.Assert.assertNotNull;
5
6 import org.junit.Test;
7
8 public class UserAccountRepositoryIntegrationTest {
9
10     private UserAccountRepository userAccountDao;
11
12     @Test
13     public void shouldFindUserAccountByLogin() {
14         final String login = "jmsanchez";
15         final UserAccount userAccount = userAccountDao.getByLogin(login);
16         assertNotNull(userAccount);
17         assertEquals("Jose Manuel Sánchez", userAccount.getName());
18     }
19 }

```

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

- no disponemos de la clase `UserAccountRepository`, que nos proporcionará acceso al repositorio de usuarios o personas,
- no tenemos la clase `UserAccount`, que tendrá las propiedades de un usuario, y
- aún no sabemos cómo configurar el servidor ldap embebido ;)

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

```

1 package com.autentia.tutoriales.spring.security.ldap;
2
3 import java.util.List;

```

```

4
5 public interface UserAccountRepository {
6
7     UserAccount getByLogin(String login);
8
9 }

```

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

```

1 package com.autentia.tutoriales.spring.security.ldap;
2
3 public class UserAccount {
4
5     private String login;
6
7     private String name;
8
9     private String email;
10
11     public String getLogin() {
12         return login;
13     }
14
15     public void setLogin(String login) {
16         this.login = login;
17     }
18
19     public String getName() {
20         return name;
21     }
22
23     public void setName(String name) {
24         this.name = name;
25     }
26
27     public String getEmail() {
28         return email;
29     }
30
31     public void setEmail(String email) {
32         this.email = email;
33     }
34
35 }

```

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

5. Implementación.

Veamos el contenido que proponemos para el repositorio de usuarios:

```

1 package com.autentia.tutorial.apacheDS.accounts;
2
3 import javax.naming.NamingEnumeration;
4 import javax.naming.NamingException;
5 import javax.naming.directory.SearchControls;
6 import javax.naming.directory.SearchResult;
7 import javax.naming.ldap.LdapContext;
8
9 public class UserAccountLDAPRepository implements UserAccountRepository{
10
11     LdapContext ldapContext;
12
13     public UserAccountLDAPRepository(LdapContext ldapContext) {
14         this.ldapContext = ldapContext;
15     }
16
17     public UserAccount getByLogin(String login) {
18         try {
19             NamingEnumeration<SearchResult> results = searchDirectory(login);
20             if (results != null){
21                 while (results.hasMoreElements()) {
22                     SearchResult searchResult = (SearchResult) results
23                         .nextElement();
24                     final UserAccount userAccount = new UserAccount();
25                     userAccount.setLogin((String) searchResult.getAttributes().get("uid");
26                     userAccount.setName((String) searchResult.getAttributes().get("cn");
27                     userAccount.setEmail((String) searchResult.getAttributes().get("mail");
28                     return userAccount;
29                 }
30             }
31             catch (NamingException e) {
32                 throw new IllegalStateException("login not found",e);
33             }
34             return null;
35         }
36
37     public NamingEnumeration<SearchResult> searchDirectory(
38         final String searchString) throws NamingException {
39         final SearchControls searchControls = new SearchControls();
40         searchControls.setSearchScope(SearchControls.SUBTREE_SCOPE);
41         return ldapContext.search("", "(uid=" + searchString + ")", searchControls);
42     }
43
44 }

```

Recibe el contexto de ldap como dependencia en el constructor, realiza la búsqueda y mapea el resultado contra el objeto de respuesta, todo ello sin usar el soporte de plantillas de Spring.

Ahora vamos a añadir el soporte del servidor LDAP embebido a nuestro test de integración.

```

1 package com.autentia.tutorial.apacheDS.accounts;
2
3 import static org.junit.Assert.assertEquals;

```

```

4  import static org.junit.Assert.assertNotNull;
5
6  import java.util.Hashtable;
7
8  import javax.naming.Context;
9  import javax.naming.NamingException;
10 import javax.naming.ldap.Control;
11 import javax.naming.ldap.InitialLdapContext;
12 import javax.naming.ldap.LdapContext;
13
14 import org.apache.directory.server.annotations.CreateLdapServer;
15 import org.apache.directory.server.annotations.CreateTransport;
16 import org.apache.directory.server.core.annotations.ApplyLdifFiles;
17 import org.apache.directory.server.core.annotations.CreateDS;
18 import org.apache.directory.server.core.annotations.CreatePartition;
19 import org.apache.directory.server.core.integ.AbstractLdapTestUnit;
20 import org.apache.directory.server.core.integ.FrameworkRunner;
21 import org.junit.Before;
22 import org.junit.Test;
23 import org.junit.runner.RunWith;
24
25 @RunWith(FrameworkRunner.class)
26 @CreateDS(allowAnonAccess = true, name = "Autentia", partitions = { @CreatePartition(name
27 @CreateLdapServer(transports = { @CreateTransport(protocol = "LDAP", port=9445) })
28 public class UserAccountRepositoryIntegrationTest extends AbstractLdapTestUnit {
29
30     private UserAccountRepository userAccountDao;
31
32     @Before
33     public void setUp() throws NamingException{
34         final LdapContext ldapContext = createLdapContext(ldapServer.getTransports()[0].get
35         userAccountDao = new UserAccountLDAPRepository(ldapContext);
36     }
37
38     @ApplyLdifFiles("autentia-identity-repository.ldif")
39     @Test
40     public void shouldFindUserAccountByLogin(){
41         final String login = "jmsanchez";
42         final UserAccount userAccount = userAccountDao.getByLogin(login);
43         assertNotNull(userAccount);
44         assertEquals("Jose Manuel Sánchez", userAccount.getName());
45     }
46
47     private LdapContext createLdapContext(String host, int port) throws NamingException {
48         final Hashtable<String, String> environment = new Hashtable<String, String>();
49         environment.put(Context.INITIAL_CONTEXT_FACTORY,
50             "com.sun.jndi.ldap.LdapCtxFactory");
51         environment.put(Context.PROVIDER_URL, "ldap://" + host + ":" + port
52             + "/" );
53
54         LdapContext ldapCtx = new InitialLdapContext(environment,
55             new Control[0]);
56         return ldapCtx;
57     }
58 }

```

Hemos hecho las siguientes modificaciones:

- nuestra clase entiende AbstractLdapTestUnit que nos va a permitir acceder al estado del servidor embebido en el entorno del test,
- FrameworkRunner.class nos permite ejecutar el test con el soporte de apacheDS,
- la anotación @CreateDS, establece las características estructurales de creación del servidor,
- la anotación @CreateLDAPServer establece las características de conectividad con el servidor,
- con la anotación @ApplyLdifFiles podemos establecer los ficheros de carga con la estructura y datos del nuestro server para el entorno de tests.

Sin más ni menos ya disponemos del soporte para comprobar nuestra lógica de acceso al directorio LDAP.

6. Referencias.

- <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=springSecurityEmbeddedLDAPServer>

7. Conclusiones.

Este tutorial responde al... pero... ¿y si no tengo el soporte de Spring? ;)

Un saludo.

Jose

jmsanchez@autentia.com

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

Por favor, vota +1 o compártelo si te pareció interesante

[Share](#) |

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

» **Registrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

PUSH THIS

no clicks

Page Pushers

Community

Help?

0 people brought clicks to this page

+

+

+

+

+

+

+

+

powered by [karmacracy](#)