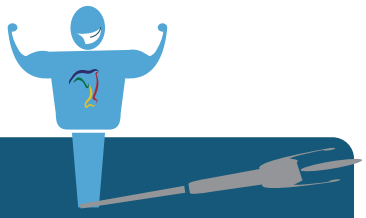
autentia

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...

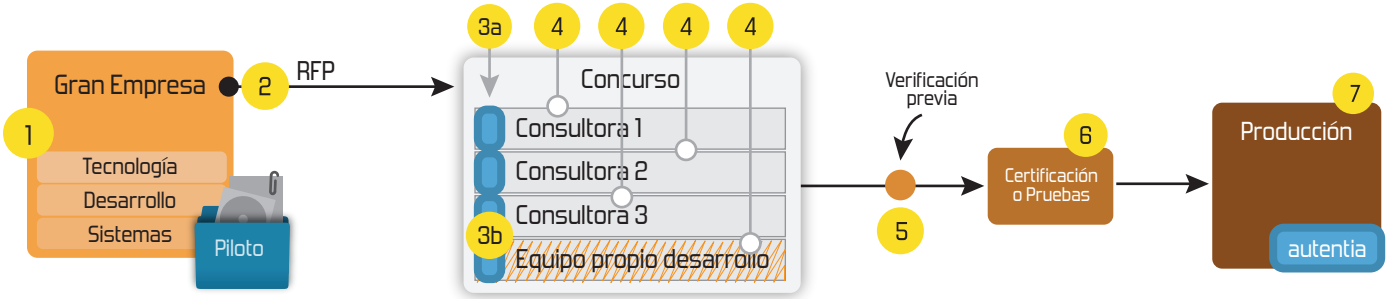
## 1. Desarrollo de componentes y proyectos a medida



## 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**

DESARROLLADO POR:
César López de Felipe Abad

Consultor tecnológico de desarrollo de proyectos informáticos.

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

Somos expertos en Java/JEE



**Fecha de publicación del tutorial: 2011-06-22** 69



# Introducción a Spring Security 3.1

## 0. Índice de contenidos.



## 1. Introducción.

En adictos ya tenemos un tutorial que hizo Enrique sobre Spring Security que podeís

encontrar aquí. En este tutorial se cubren básicamente los mismos aspectos utilizando Spring Security 3, utilizaremos el tutorial-sample que viene con Spring Security y seguiremos los pasos que explican en su tutorial en inglés http://static.springsource.org/spring-security/site/tutorial.html.

Toda la configuración necesaria la vamos a realizar a través de archivos xml por lo que no hace falta bajarse los fuentes del tutorial, con descargarnos el war, descomprimirlo y ponerlo en nuestro directorio webapp de tomcat podremos ejecutar la aplicación e ir siguiendo los pasos para añadirle seguridad.

En este tutorial vamos a ver como autenticar y autorizar usuarios en base a unos logins, contraseñas y roles que tendremos en un archivo de configuración xml. También veremos como usar la encriptación de los passwords. Y en próximos tutoriales iremos ampliando las posibilidades de configuración que ofrece Spring Security.

Para usar Spring Security sólo es necesario conocer como funcionan los archivos de configuración de Spring y las inyecciones de dependencias.

## 2. Entorno.

- Hardware: MacBookPro8,2 (2 GHz Intel Core i7, 4GB DDR3 SDRAM).
- AMD Radeon HD 6490M 256MB.
- Sistema Operativo: Mac OS X Snow Leopard 10.6.7.
- Servidor: Apache Tomcat 6.0.32
- Spring Security 3.1.0.RC2

## 3. Descargar el war del tutorial de Spring Security y colocar la aplicación en tomcat.

Lo primero que tenemos que hacer es bajarnos la última versión de Spring Security desde su página web http://static.springsource.org/spring-security/site/downloads.html, ahora mismo es la versión 3.1.0.RC2, esta versión no es cien por cien estable pero no se anticipan muchos cambios con respecto a la próxima release, requiere como mínimo Spring 3.0.5 y Java 5. Podeís rellenar el formulario o hacer click en la parte de abajo del mismo para continuar a la página de descargas sin rellenarlo.

Una vez descargado, descomprimís el zip y dentro del mismo encontrareís un war que se llamará spring-security-samples-tutorial-3.1.0.RC2, este será el proyecto web que utilizaremos. Lo descomprimimos y renombramos la carpeta a algo más sencillo como tutorial-spring-security. Copiamos esa carpeta en el directorio webapp de nuestro tomcat, levantamos tomcat y ya podemos acceder a http://localhost:8080/tutorial-spring-security/.

## 4. Añadiendo seguridad a la web.

Ahora mismo el proyecto tiene la seguridad activada y si intentamos acceder a Secure page o Extremely secure page nos pedirá autenticarnos, pasará lo mismo si dentro de list accounts intentamos modificar la cantidad con los links de -$20 -$5 +$5 +$20.

Lo primero que vamos a hacer es desactivar la seguridad, para ello tenemos que reemplazar el contenido del web.xml por este:

```
01 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemalocation="http://java.sun.com/xml/ns/javaee
02                 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
03
04     <display-name>Spring Security Tutorial
   Application</display-name>
```

Últimas Noticias

XVII Charla Autentia - Grails - Vídeos y Material

¡¡¡ 15 millones de descargas de tutoriales !!!

XVII Charla Autentia - Grails

Charla en WhyFLOSS en el IE: la ppt

Charla en TheEvnt: La Technicienta, de programador a empresario, la ppt

Histórico de NOTICIAS

Últimos Tutoriales

Implementando SSO con CAS: ejemplo práctico

Como desarrollar un plugin para Eclipse

Técnica del Time-Lapse

Incluir Gadgets en Liferay 6.0.5: Cómo añadir Gadgets de forma sencilla

Crear un paginador utilizando JSTL Core

Últimos Tutoriales del Autor

Crear un juego

```
05
06      <!--
07        - Location of the XML file that defines the root
application context
08        - Applied by ContextLoaderListener.
09       -->
10      <context-param>
11          <param-name>contextConfigLocation</param-name>
12          <param-value>
13              classpath:applicationContext-business.xml
14          </param-value>
15      </context-param>
16
17      <listener>
18          <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
19      </listener>
20      <!--
21        - Provides core MVC application controller. See
bank-servlet.xml.
22       -->
23      <servlet>
24          <servlet-name>bank</servlet-name>
25          <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
26          <load-on-startup>1</load-on-startup>
27      </servlet>
28
29      <servlet-mapping>
30          <servlet-name>bank</servlet-name>
31          <url-pattern>*.html</url-pattern>
32      </servlet-mapping>
33
34      <welcome-file-list>
35          <welcome-file>index.jsp</welcome-file>
36      </welcome-file-list>
37  </web-app>
```

Si intentais acceder ahora a cualquiera de las partes de la aplicación que antes nos pedían autenticarnos vereís que podeís acceder sin problemas, aunque el enlace de logout y la página index.jsp han dejado de funcionar.

Vamos a añadir la configuración de Spring Security. Lo más habitual es crear un application context separado del resto de los archivos de configuración de la aplicación. Vamos a crear este archivo que llamaremos security-app-context.xml dentro del WEB-INF y le añadimos este contenido:

```
01  <beans:beans xmlns="http://www.springframework.org/schema
/security" xmlns:beans="http://www.springframework.org/schema
/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemalocation="http://www.springframework.org/schema/beans
02                      http://www.springframework.org/schema/beans
/spring-beans-3.0.xsd
03                      http://www.springframework.org/schema
/security
04                      http://www.springframework.org/schema
/security/spring-security-3.1.xsd">
05
06      <http use-expressions="true">
07          <intercept-url pattern="/**" access="permitAll">
08          <form-login>
09      </form-login></intercept-url></http>
10
11      <authentication-manager>
12          <authentication-provider>
13              <user-service>
14                  <user name="rod" password="koala"
authorities="supervisor, teller, user">
15                  <user name="dianne" password="emu"
authorities="teller, user">
16                  <user name="scott" password="wombat"
authorities="user">
17                  <user name="peter" password="opal"
authorities="user">
18              </user></user></user></user></user-service>
19          </authentication-provider>
```

```
    </authentication-manager>
</beans:beans>
```

Estamos usando el namespace de Spring Security para crear un configuración simple. En el bloque http, estamos diciendo que las expresiones para el control de acceso estan activadas y que usaremos un login basado en un formulario. Y en el intercept-url estamos diciendo que todo lo que cumpla el patron /** (el raíz y todos sus subdirectorios) es accesible a todo el mundo "permitAll". En el bloque del authentication-manager estamos definiendo una serie de usuarios en memoria con sus contraseñas y sus roles, que para este tutorial es suficiente. En una aplicación real se pueden utilizar usuarios y roles de una base de datos, de un servidor LDAP o integrarse con sistemas de single sign-on (como por ejemplo OpenID).

Una vez configurado el archivo hay que añadirlo al context-param del web.xml

```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>
       classpath:applicationContext-business.xml
       /WEB-INF/security-app-context.xml
   </param-value>
</context-param>
```

Y añadimos debajo lo siguiente:

```
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>

<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Si tomcat no reinicia la aplicación al guardar el web.xml, habrá que reiniciarlo. Ahora mismo podemos acceder a http://localhost:8080/tutorial-spring-security/index.jsp, y a las páginas secure page y extremely secure page (esto dice el tutorial de spring, aunque a mi ahora mismo me da un error porque no puede acceder a la propiedad username del objeto principal de Spring Security).

Ya estamos listos para añadir algo de seguridad. Vamos a ir añadiendola para cumplir la siguiente funcionalidad:

- 1. Debes estar autenticado para acceder a la "account list"
- 2. Las páginas del directorio "/secure" sólo serán visibles para usuarios autenticados.
- 3. Las páginas del directorio "/secure/extreme" sólo serán visibles para los supervisores.
- 4. Sólo los usuarios con los roles de "teller" y "supervisor" podrán retirar y depositar dinero.

Suele ser una buena práctica negar el acceso por defecto además de asegurar las recursos que necesitamos. Vamos a añadir unos cuantos patrones más para interceptar urls. Modificamos el bloque http por el siguiente:

```
<http use-expressions="true">
    <intercept-url pattern="/index.jsp" access="permitAll">
    <intercept-url pattern="/secure/**" access="isAuthenticated()">
    <intercept-url pattern="/secure/extreme/**" access="hasRole('supervisor')">
    <intercept-url pattern="/listAccounts.html" access="isAuthenticated()">
    <intercept-url pattern="/post.html" access="hasAnyRole('supervisor','teller')">
    <intercept-url pattern="/**" access="denyAll">
    <form-login></form-login></intercept-url></intercept-url></intercept-url></intercept-url></intercept-url></http>
```

Con esta configuración cualquiera podrá acceder al index.jsp, los usuarios

autenticados podrán acceder a la carpeta secure y sus subdirectorios así como a /listAccounts.html, los usuarios con el rol "supervisor" podrán acceder a /secure /extreme y sus subdirectorios, cualquiera con el rol de "supervisor" o "teller" podrá acceder a la /post.html y por último se negará el acceso al resto de urls.

Las condiciones se evaluan en orden, por lo que los patrones más especificos deben ir primero, ahora mismo si intentamos acceder a otra página que no sea la de index.jsp nos redigirá a la pantalla de login. Es posible que tengaís que borrar la cache del navegador para que los cambios se vean reflejados.

Podeís probar la aplicación usando los usuarios y las contraseñas del archivo de configuración, dependiendo de los roles que tenga podrá acceder a unos recursos o a otros. Podeís ver también que después de hacer el login se carga directamente la página a la que se estaba navegando, esto lo gestiona automáticamente Spring Security.

Puede interesar que recursos estáticos como las hojas de estilo no se vean afectadas por los filtros de Spring Security, para lo que podemos añadir un bloque adicional de http que sólo se aplique a un patrón específico, este bloque deberá estar antes del bloque http que ya tenemos configurado. Si no le ponemos ningún patron se aplicará para todas las peticiones.

```
1      <http pattern="/static/**" secured="false">
2  </http>
```

Con este patrón haremos que no se aplique seguridad a los recursos dentro de la carpeta static y sus subdirectorios.

Si habeís intentado usar el link de logout para desconectaros habreís visto que os dice que el acceso esta denegado. Para habilitar la pantalla de logout basta con añadir al final del segundo bloque http la linea:

```
1      <logout>
2  </logout>
```

Con esto tendriamos habilitada la posibilidad de desconectarnos.

Para finalizar con el tutorial vamos a incluir el encriptado de contraseñas, una practica altamente recomendable y que con Spring Security es bastante sencillo de usar. En nuestro caso basta con modificar la parte correspondiente al authentication manager y añadir el bean que se encargara de realizar el cifrado de la contraseña.

```
01 <beans:bean id="encoder"
   class="org.springframework.security.crypto.password.StandardPasswordEncoder">
02
03 <authentication-manager>
04   <authentication-provider>
05     <password-encoder ref="encoder">
06     <user-service>
07       <user name="rod"
   password="864acff7515e4e419d4266e474ea14a889dce340784038b704a30453e01245eed374f881f3df8e1e"
   authorities="supervisor, teller, user">
08       <user name="dianne"
   password="9992e040d32b6a688ff45b6e53fd0f5f1689c754ecf638cce5f09aa57a68be3c6dae699091e58324"
   authorities="teller, user">
09       <user name="scott"
   password="ab8d9744fa4dd5cee6eb692406fd29564267bad7c606837a70c44583b72e5684ec5f55c9dea869a5"
   authorities="user">
10       <user name="peter"
   password="e446d30fcb00dc48d7e9fac49c2fec6a945171702e6822e1ec48f1ac1407902759fe30ed66a068df"
   authorities="user">
11       </user></user></user></user></user-service>
12   </password-encoder></authentication-provider>
13 </authentication-manager>
14 </beans:bean>
```

Las passwords siguen siendo las mismas de antes lo único que ahora estan encriptadas utilizando la clase StandardPasswordEncoder.

La implementación que nos proporciona el StandardPasswordEncoder aplica 1024 iteraciones del algoritmo SHA-256 combinada con un salt aleatorio de 8-byte. Además se puede utilizar una clave para toda la aplicación que se combinará también con la password y el salt autogenerado. Esto hace que las passwords sean menos

vulnerables a un ataque por fuerza bruta.

Con esto finalizaría este tutorial, en próximos tutoriales seguiremos explorando las posibilidades que nos ofrece Spring Security.

## 5. Referencias

http://static.springsource.org/spring-security/site/

http://static.springsource.org/spring-security/site/tutorial.html

http://static.springsource.org/spring-security/site/articles.html

http://www.adictosaltrabajo.com/tutoriales
/tutoriales.php?pagina=utilizaciondegruposenspringsecurity

## 6. Conclusiones.

En este tutorial hemos seguido de forma sencilla el tutorial básico de Spring Security y hemos visto que se puede implementar seguridad a través de urls y de roles de una forma sencilla a través de archivos xml.

En próximos tutoriales seguiremos explorando las posibilidades que nos ofrece Spring Security. Espero que os haya gustado el tutorial y si quereís hacer algún comentario, sugerencia o preguntar alguna duda podeís hacerlo en la zona de comentarios.

Un saludo.

César López.