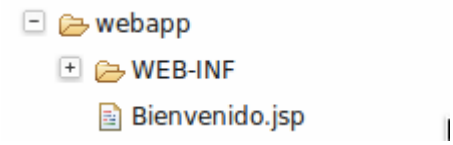
Quizas una de las partes mas utilizadas y que mas dudas genera en Spring Framework es el framework Spring Security ya que a veces parece que es inmenso y muchas personas no son expertas en seguridad. Vamos a dedicar algunos post a hablar un poco de el y de los conceptos principales que aborda.

Para ello en un primer momento deberemos instalar el framework. Vamos a crear un proyecto Maven con Eclipse que nos permita añadir las siguientes dependencias que son las necesarias para que Spring Security funcione.(si tienes dudas sobre como usar maven puedes revisar los siguientes articulos del blog)

```
<dependencies>
<dependency>
 <groupId>org.springframework.security</groupId>
 <artifactId>spring-security-web</artifactId>
 <version>3.2.2.RELEASE</version>
 </dependency>
 <dependency>
 <groupId>org.springframework.security</groupId>
 <artifactId>spring-security-config</artifactId>
 <version>3.2.2.RELEASE</version>
 </dependency>
 <dependency>
 <groupId>commons-logging</groupId>
 <artifactId>commons-logging</artifactId>
 <version>1.1.3</version>
 </dependency>
 </dependencies>
```

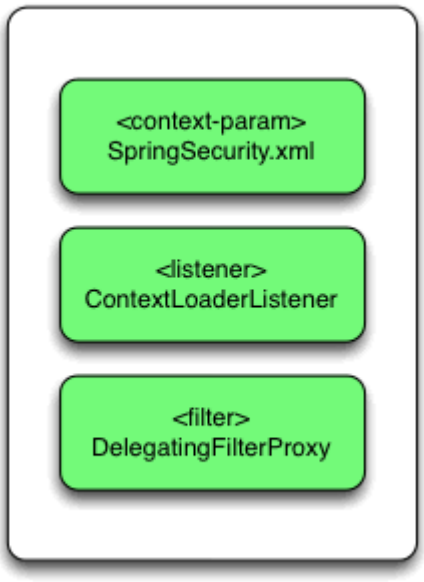
Una vez que tenemos configuradas las dependencias . Vamos a desarrollar la aplicación web

mas básica existente que solo incluye una pagina JSP.

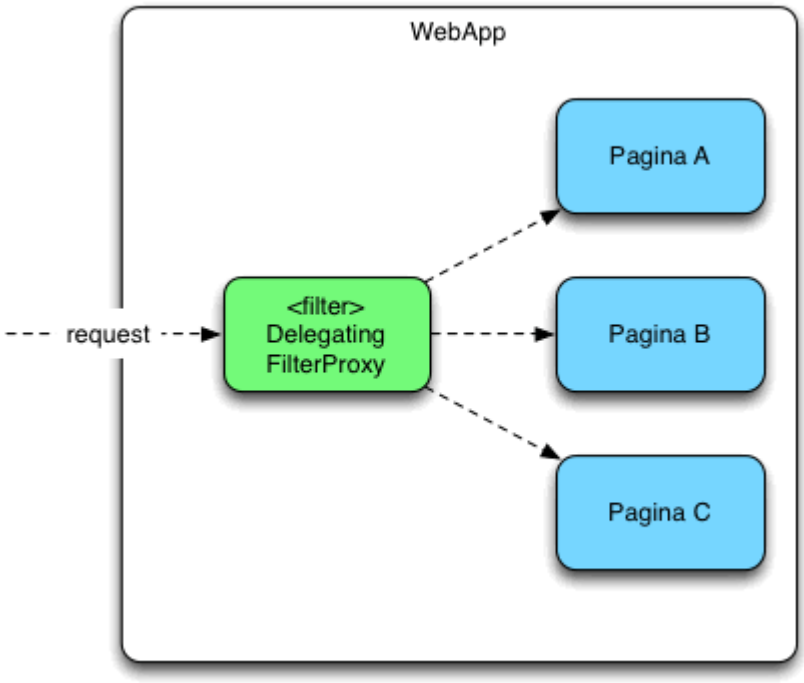


Spring Security (Configuración)

Ya tenemos la aplicación construida y deberemos configurarla para que quede protegida por Spring Security . Para ello el primer paso que debemos realizar es dar de alta en el fichero web.xml la ruta en donde tenemos ubicado el fichero de configuración de Spring (usando un context param) . El siguiente paso es declarar un listener que nos inicialice el framework y por último un filtro que protega toda la aplicación de accesos no permitidos y delegue en Spring Framework todas las operativas de seguridad. Así pues el web.xml tendra el siguiente contenido.



Una vez tenemos configurado el fichero web.xml ,el filtro de SpringSecurity se encargará de bloquear el acceso a toda la aplicación.



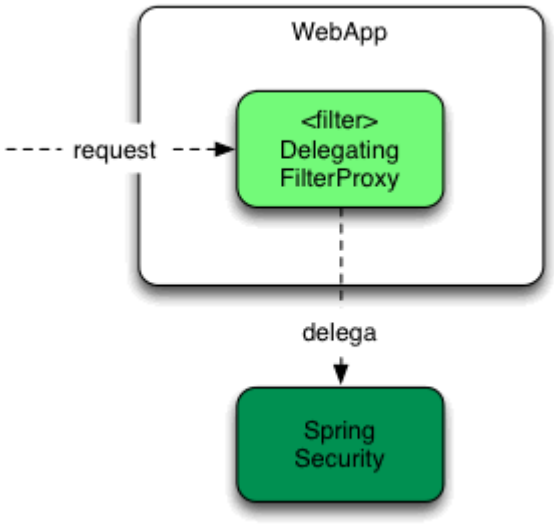
Vamos a ver el código fuente del web.xml para clarificar dudas:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
 <display-name>web01</display-name>
<!-- ruta fichero -->
<context-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>
 /WEB-INF/springSecurity.xml
```

```
 </param-value>
</context-param>
<!-- listener carga Spring-->
<listener>
 <listener-class>
 org.springframework.web.context.ContextLoaderListener
 </listener-class>
</listener>
<!-- filtro de Spring security-->
<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>
</web-app>
```

SpringSecurity.xml

Acabamos de configurar el framework Spring ,es momento de ver el contenido del fichero springsecurity.xml . Este fichero es al cual el filtro de Spring delega para gestionar la seguridad.



En este caso hemos elegido un fichero muy sencillo para no complicar las cosas :

```
<?xml version="1.0" encoding="UTF-8"?>
<bean:beans xmlns:bean="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.springframework.org/schema/security"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.2.xsd
">
 <http auto-config="true">
 <intercept-url pattern="/**" access="ROLE_Usuario" />
 </http>
 <authentication-manager>
 <authentication-provider>
 <user-service>
```

```
 <user name="manuel" password="1234" authorities="ROLE_Usuario" />
 </user-service>
 </authentication-provider>
 </authentication-manager>
</bean:beans>
```

El fichero define varios conceptos fundamentales

A) El grupo de recursos protegidos (URLs)

```
 <http auto-config="true">
 <intercept-url pattern="/**" access="ROLE_Usuario" />
 </http>
```

En este caso esta toda la aplicación protegida /** y solo se permite el acceso al role “ROLE_Usuario”.

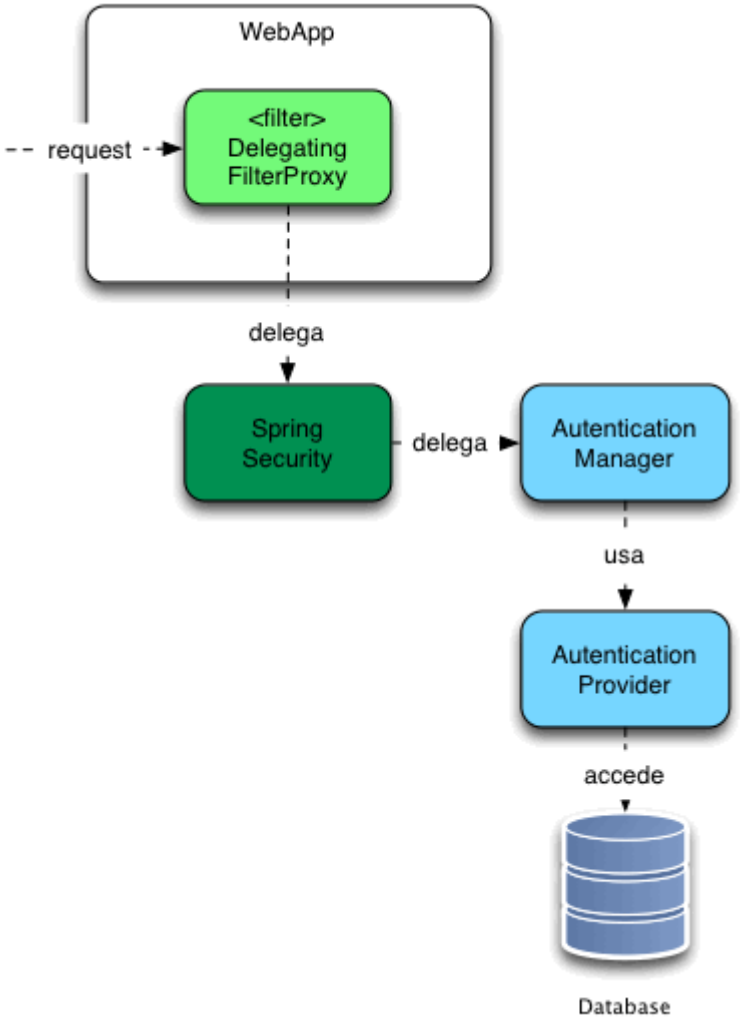
B) AutenticationManager o gestor de autentificación que decide cuando un usuario es válido. Este gestor esta intimamente relacionado con el concepto de AutenticationProvider o proveedor de autentificación .

```
<authentication-manager>
 <authentication-provider>
 <user-service>
 <user name="manuel" password="1234" authorities="ROLE_Usuario" />
 </user-service>
 </authentication-provider>
```

C) Este último el AutenticationProvider define la forma en la que un usuario se ha de validar . Puede ser contra una base de datos , puede ser contra un Ldap, puede ser contra un fichero o puede personalizarse.En nuestro caso estamos utilizando un servicio en memoria que solo permite el acceso al siguiente usuario.

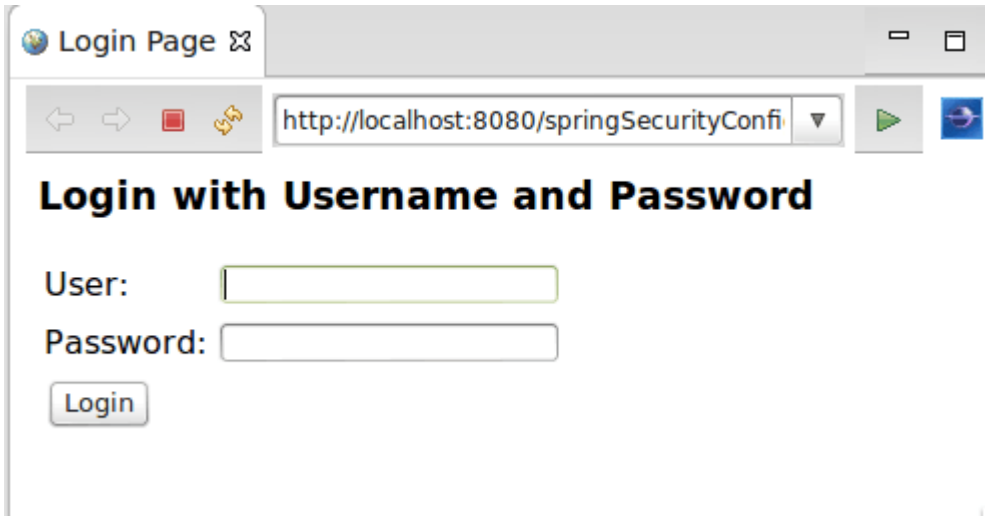
```
<user name="manuel" password="1234" authorities="ROLE_Usuario" />
```

El siguiente diagrama clarifica la relación entre los elementos.



Realizada esta operación ejecutamos “maven package” y nos empaquetará la aplicación de

tal forma que si la desplegamos en un servidor (Tomcat por ejemplo) e intentamos acceder a la página de bienvenidos ,Spring Security nos bloqueará el acceso y mostrará la siguiente pantalla.



Introducimos de usuario manuel y de clave 1234 y el framework nos dará acceso a la página protegida.

Acabamos de construir el ejemplo de hola mundo con Spring Security.