La necesidad de usar Spring Security Annotation y eliminar los XML esta creciendo . La mayoría de nosotros usamos Spring 4.X a la espera del salto que depare la versión 5 con flux etc. Sin embargo a veces es difícil encontrar documentación clara de como configurar el framework a nivel de seguridad utilizando anotaciones y eliminando los XML. Vamos a ver como podemos hacerlo y cuales son los pasos a seguir. Lo primero que deberemos hacer es configurar las dependencias de Maven necesarias.

```
&amp;lt;dependencies&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;spring-
context&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;4.3.10.RELEASE&amp;lt;/version&amp;gt;
                &amp;lt;/dependency&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;spring-
webmvc&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;4.3.10.RELEASE&amp;lt;/version&amp;gt;
                &amp;lt;/dependency&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;javax.servlet&amp;lt;/groupId&amp;gt;
```

<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
</dependency>
<dependency>
<groupId>javax.servlet.jsp</groupId>
<artifactId>javax.servlet.jsp-api</artifactId>
<version>2.3.1</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-web</artifactId>
<version>4.2.3.RELEASE</version>
</dependency>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-config</artifactId>
<version>4.2.3.RELEASE</version>
</dependency>

```
&amp;lt;/dependencies&amp;gt;
```

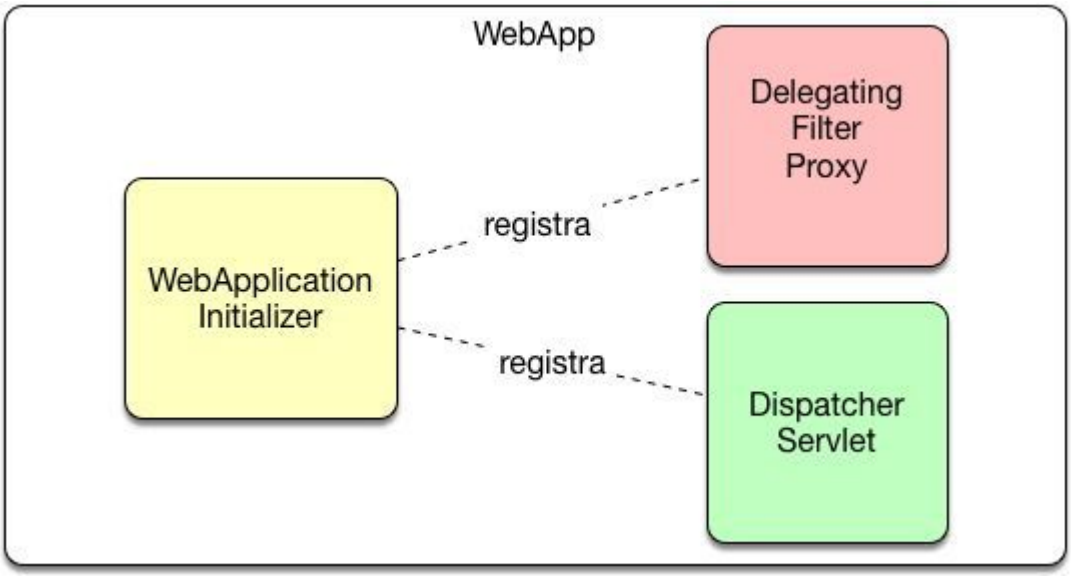
Spring Security Annotation

El segundo paso es crear un WebInitializer. ¿Qué es un un WebInitializer? . Se trata de una clase que se ejecuta al arrancar nuestra aplicación web y puede abordar tareas adicionales que la aplicación necesita como por ejemplo registrar Servlets o Filtros. Veamos su código:

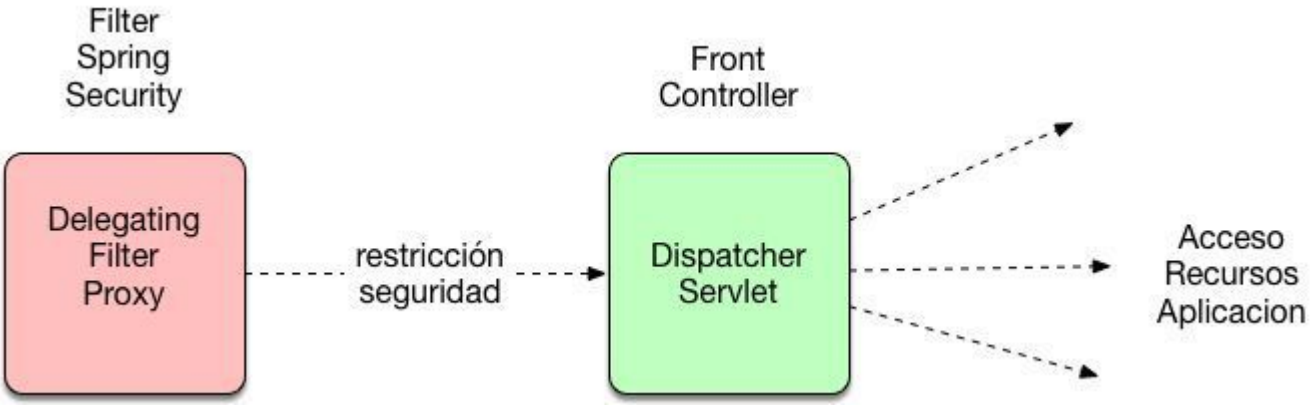
```
package com.arquitecturajava.inicializador;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;
import
org.springframework.web.context.support.AnnotationConfigWebApplication
Context;
import org.springframework.web.filter.DelegatingFilterProxy;
import org.springframework.web.servlet.DispatcherServlet;

import com.arquitecturajava.SpringConfiguracion;

public class SpringInicializador  implements
WebApplicationInitializer{

        public void onStartup(ServletContext container) throws
ServletException {
```

```
                AnnotationConfigWebApplicationContext ctx = new
AnnotationConfigWebApplicationContext();
             ctx.register(SpringConfiguracion.class);
             ctx.setServletContext(container);

             ServletRegistration.Dynamic servlet =
container.addServlet("dispatcher", new DispatcherServlet(ctx));
             servlet.setLoadOnStartup(1);
             servlet.addMapping("/");

             container.addFilter("springSecurityFilterChain", new
DelegatingFilterProxy("springSecurityFilterChain"))
          .addMappingForUrlPatterns(null, false, "/*");
        }

        }
```

El WebInitializer , se encarga de registrar el dispatcher Servlet de Spring Framework que es el que hace el rol de FrontController y el DelegatingFilterProxy que es el filtro que habilita Spring Security en nuestra aplicación.



Una vez registrados el filtro protege el acceso al DispatcherServlet y por consecuencia al resto de la aplicación.



Una vez tenemos estos pasos realizados el siguiente paso es crear un controlador que tenga un par de urls de acceso.

```
package com.arquitecturajava.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
```

```
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class ControladorHola {

	@RequestMapping("/hola")
	public String hola(ModelMap model) {

		model.addAttribute("mensaje", "hola desde spring
mvc");
		return "hola";

	}

	@RequestMapping("/adios")
	public String adios(ModelMap model) {

		model.addAttribute("mensaje", "adios desde spring
mvc");
		return "adios";

	}

}
```

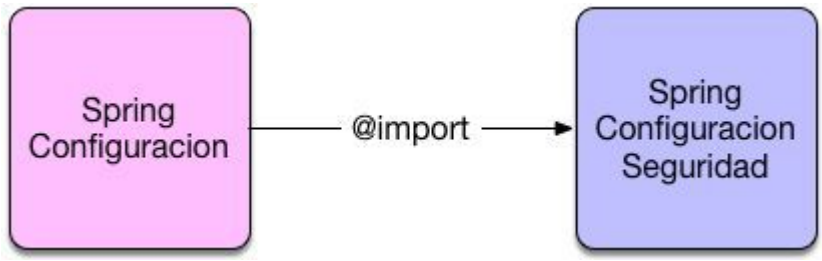
Hecho esto nos queda crear un fichero de configuración de Spring que registre un ViewResolver y de de alta nuestro controlador a través de la anotación component scan.

```
package com.arquitecturajava;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Import;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@EnableWebMvc
@Configuration
@ComponentScan("com.arquitecturajava.*")
@Import({SpringConfiguracionSeguridad.class})
public class SpringConfiguracion {

        @Bean
               public ViewResolver viewResolver() {
                System.out.println("llega");
                       InternalResourceViewResolver viewResolver =
new InternalResourceViewResolver();
                       viewResolver.setViewClass(JstlView.class);
                       viewResolver.setPrefix("/WEB-INF/vistas/");
                       viewResolver.setSuffix(".jsp");

                       return viewResolver;
               }

}
```

El siguiente paso es construir los ficheros JSP que van a ser nuestras vistas(ambos ficheros son iguales):

```
&amp;lt;%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%&amp;gt;
&amp;lt;!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"&amp;gt;
&amp;lt;html&amp;gt;
&amp;lt;head&amp;gt;
&amp;lt;meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"&amp;gt;
&amp;lt;title&amp;gt;Insert title here&amp;lt;/title&amp;gt;
&amp;lt;/head&amp;gt;
&amp;lt;body&amp;gt;
${mensaje }
&amp;lt;/body&amp;gt;
&amp;lt;/html&amp;gt;
```

Nos queda de configurar el fichero concreto de Spring Security Annotation (SpringConfiguracionSeguridad.class) , que es el encargado de definir las url protegidas de acceso y el cual tenemos referenciado desde el fichero de configuración principal con @import.



```
package com.arquitecturajava;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders
.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.Enabl
eWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSe
curityConfigurerAdapter;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@EnableWebSecurity
public class SpringConfiguracionSeguridad  extends
WebSecurityConfigurerAdapter{

        @Override
        protected void configure(HttpSecurity http) throws Exception {
```

```
http.authorizeRequests().antMatchers("/**").hasRole("BASICO").and().fo
rmLogin();
                super.configure(http);
        }

         @Autowired
            public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
                auth
                    .inMemoryAuthentication()
.withUser("cecilio").password("miclave").roles("BASICO");
            }

}
```

En este caso protegemos la aplicación completa y únicamente registramos un usuario con rol básico en memoria para comprobar el acceso. Es momento de solicitar una de las dos páginas configuradas en el navegador http://localhost:8080/springsecurity/hola . Una vez solicitada la url , seremos redireccionados al login de la aplicación con el formulario por defecto de Spring Security.

Introducimos el nombre y la clave del usuario y accederemos al recurso solicitado.

Acabamos de configurar Spring 4 y Spring Security usando anotaciones.

Otros artículos relacionados:

1. Utilizando Spring MVC configuration annotation
2. Spring Security (I) configuracion
3. Spring MVC @RequestMapping
4. Spring Security Referencia