Spring Security

Spring Security

- Framework para autenticación y control de acceso
  - Es el estándar de seguridad para aplicaciones hechas en Spring
- Características
  - Soporte completo y extensible para autenticación y autorización
  - Protección contra ataques “session fixation”, “clickjacking”, “cross site request forgery”...
  - Integración con Servlet API
  - Integración con Spring MVC

Spring Security

- Lo primero que se debe hacer: crear el “Spring Security Java Configuration”
- Crea un Servlet Filter conocido como “springSecurityFilterChain”, responsable de toda la seguridad:
  - Protección de las URL de la aplicación
  - Validación de usuario y contraseña
  - Redirección a formulario de login

Spring Security

```java
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.*;
import org.springframework.security.config.annotation.authentication.builders.*;
import org.springframework.security.config.annotation.web.configuration.*;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
            .inMemoryAuthentication()
                .withUser("user").password("password").roles("USER");
    }
}
```

Spring Security

- El SecurityConfig:
  - Solicita autenticación para cualquier URL de la aplicación
  - Genera un formulario de login
  - Permite el acceso del usuario “user” con password “password”
  - Prevención para ataque CSRF y Session Fixation
  - Integración con “Security Header”: HTTP Strict Transport Security, X-Content-Type-Options, Cache-Control, X-XSS-Protection, etc

Spring Security

- El siguiente paso es “registrar” el springSecurityFilterChain en el WAR
  - Spring provee una clase base llamada “AbstractSecurityWebApplicationInitializer”, la cual asegura que el “springSecurityFilterChain” será registrado
- Si no se está usando Spring o Spring MVC, es necesario pasar el SecurityConfig a una superclase

Spring Security

```
import org.springframework.security.web.context.*;

public class SecurityWebApplicationInitializer
      extends AbstractSecurityWebApplicationInitializer {

    public SecurityWebApplicationInitializer() {
        super(SecurityConfig.class);
    }
}
```

Spring Security

- SecurityWebApplicationInitializers:
  - Registrará automáticamente el filtro springSecurityFilterChain para cualquier URL de la aplicación
  - Agregará un “ContextLoaderListener” que cargará el “SecurityConfig”

Spring Security

- El SecurityConfig contiene información sobre cómo se autentican los usuarios
  - No dice si todos los usuarios deben ser autenticados
  - No dice si deben ser autenticados con un formulario
- Hay una configuración “default” provista por WebSecurityConfigAdapter

```
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .and()
        .httpBasic();
}
```

Spring Security

- La configuración default:
  - Asegura que cualquier solicitud a la aplicación web requiere autenticación de usuario
  - Permite a los usuario autenticarse con un formulario de login
  - Permite a los usuarios autenticarse con “HTTP Basic Authentication”
  - Configuración es similar a:

```
<http use-expressions="true">
    <intercept-url pattern="/**" access="authenticated"/>
    <form-login />
    <http-basic />
</http>
```

Spring Security

- Se puede mejorar la configuración:

```
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .permitAll();
}
```

  - Indica una página de login
  - Todos los usuarios pueden llegan a la página de login

```
<c:url value="/login" var="loginUrl"/>
<form action="${loginUrl}" method="post">
    <c:if test="${param.error != null}">
        <p>
            Invalid username and password.
        </p>
    </c:if>
    <c:if test="${param.logout != null}">
        <p>
            You have been logged out.
        </p>
    </c:if>
    <p>
        <label for="username">Username</label>
        <input type="text" id="username" name="username"/>
    </p>
    <p>
        <label for="password">Password</label>
        <input type="password" id="password" name="password"/>
    </p>
    <input type="hidden"
        name="${_csrf.parameterName}"
        value="${_csrf.token}"/>
    <button type="submit" class="btn">Log in</button>
</form>
```

Spring Security

- Se pueden definir distintas autenticaciones para diferentes URL

```
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .antMatchers("/resources/**", "/signup", "/about").permitAll()
            .antMatchers("/admin/**").hasRole("ADMIN")
            .antMatchers("/db/**").access("hasRole('ROLE_ADMIN') and hasRole('ROLE_DBA')")
            .anyRequest().authenticated()
            .and()
        // ...
        .formLogin();
}
```