Partner ORACLE UNIVERSITY

Certifica TIC Beyond Certification

INICIO

Desarrollo de Aplicaciones Empresariales con Spring Framework Core 5

ISC. Ivan Venor García Baños

ATC Advanced Training Consultants

SCRUMstudy Authorized Training Partner

Agenda

1. Presentación
2. Objetivos
3. Contenido
4. Despedida

3. Contenido

i. Introducción a Spring Framework
ii. Spring Core
iii. Spring AOP
iv. Spring JDBC – Transaction
v. Spring ORM – Hibernate 5
vi. Spring Data JPA
vii. Fundamentos Spring MVC y Spring REST
viii. Fundamentos Spring Security
ix. Seguridad en Servicios REST
x. Introducción Spring Boot

vii. Fundamentos Spring Security

vii. Fundamentos Spring Security

vii.i Introducción

a. ¿Qué es Spring Security?

b. Módulos Spring Security

vii.ii Spring Security Web MVC

a. Dependencias

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml

b. Configuración Spring Security Context

c. Formulario Login / Logout

d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.i Introducción

Objetivos de la lección

vii.i Introducción

- Comprender qué es Spring Security.
- Analizar los diferentes escenarios donde es posible aplicar seguridad con Spring Security.
- Conocer a grandes rasgos los módulos de Spring Security.

vii.i Introducción

a. ¿Qué es Spring Security?

b. Módulos Spring Security

vii.i Introducción (a)

- ¿Qué es Spring Security?

- Spring Security es un framework poderoso, robusto y altamente personalizable para controlar autenticación y autorización (control de acceso) en aplicaciones Java EE.

- Es el framework *de-facto* para la implementación de seguridad en aplicaciones basadas en Spring Framework.

vii.i Introducción (b)

- ¿Qué es Spring Security?

- El verdadero poder de Spring Security se encuentra en la facilidad de integrar seguridad a aplicaciones Java EE, el mecanismo de extensibilidad que proporciona y la fácil configuración del mismo.

vii.i Introducción (c)

- ¿Qué es Spring Security? (a)

- Spring Security 3.x provee las siguientes funcionalidades:
  - Autenticación y autorización
  - Autenticación mediante formularios
  - Autenticación LDAP
  - Autenticación OpenID
  - Implementación SSO
  - Implementación Cross-Site Request Forgery (CSRF)
  - Implementación "remember-me"
  - Implementación ACL
  - Implementación "channel security" (HTTP/HTTPS)
  - Soporte para JAAS
  - Soporte para Flow Authorization (Spring WebFlow)

vii.i Introducción (d)

- ¿Qué es Spring Security? (b)

- Spring Security 3.x provee las siguientes funcionalidades:
  - Implementación WS-Security (Spring Web Services)
  - Configuración mediante XML o anotaciones.

vii.i Introducción (e)

- ¿Qué es Spring Security? (c)

- Spring Security 4.x provee las siguientes funcionalidades:
  - Soporte WebSocket Security
  - Integración Spring Data
  - CSRF Token Argument Resolver

vii.i Introducción (f)

- ¿Qué es Spring Security?

- Spring Security provee dos niveles de autorización:
  - Autorización a nivel de método (Method level authorization)
  - Autorización a nivel de URL (URL level authorization).

- La autorización a nivel de método requiere Spring AOP y Spring Security Aspects.

vii.i Introducción (g)

- ¿Qué es Spring Security?

- Ventajas
  - Open Source.
  - Flexible, fácil de desarrollar y probar.
  - Permite implementar seguridad declarativa.
  - Facilidad de extensión.
  - Facilidad de mantenibilidad.
  - Aprovechamiento de DI y AOP.
  - Implementación de Seguridad mediante alta cohesión y bajo acoplamiento

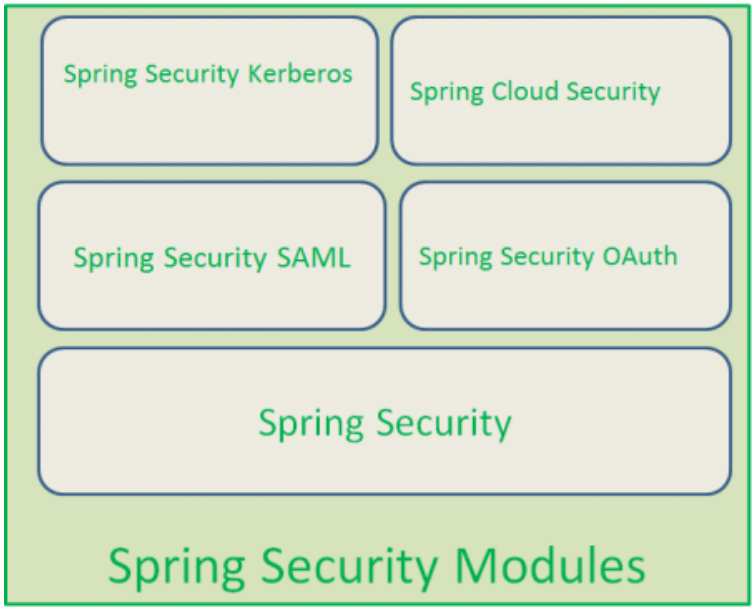
vii.i Introducción

a. ¿Qué es Spring Security?

b. Módulos Spring Security

vii.i Introducción (a)

- Módulos Spring Security



vii.i Introducción (b)

- Módulos Spring Security

- Spring LDAP

- Módulo que permite facilitar operaciones LDAP *(Lightweight Directory Access Protocol)* al estilo JdbcTemplate.

vii.i Introducción (c)

- Módulos Spring Security

- Spring Security OAuth *(Open Authorization)*

- Provee soporte de integración para aplicaciones Java EE con OAuth y OAuth2 mediante configuración de beans.

vii.i Introducción (d)

- Módulos Spring Security

- Spring Security SAML *(Security Assertion Markup Language)*

- Provee soporte de integración para proveedores SAML para la implementación de *web single sing-on, single logout*, entre otras funcionalidades.

vii.i Introducción (e)

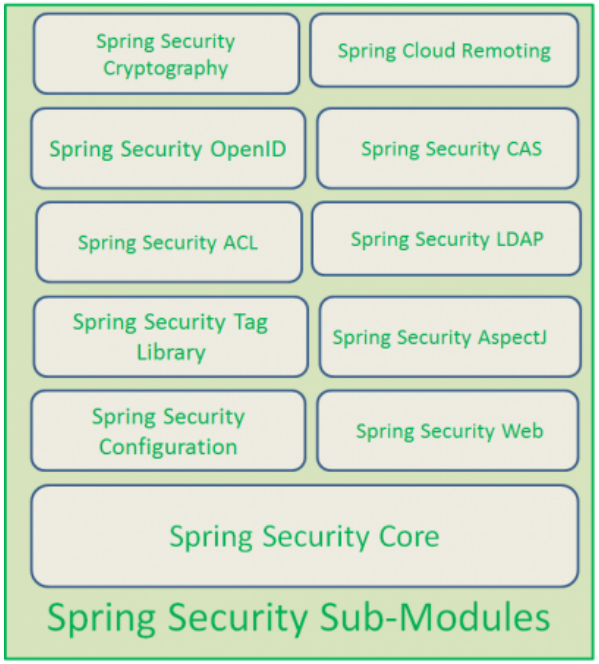
- Módulos Spring Security

- Spring Security Kerberos

- Provee soporte de integración con Kerberos, permitiendo la autenticación de usuarios simplemente mediante la petición a una URL especifica, sin necesidad de enviar a través de la web un usuario y contraseña y, sin instalar algún software adicional.

vii.i Introducción (f)

- Sub-Módulos Spring Security



vii.i Introducción (g)

- Sub-Módulos Spring Security
  - Spring Security Core
  - Spring Security Configuration
  - Spring Security Web
  - Spring Security Tag Library
  - Spring Security AspectJ
  - Spring Security ACL
  - Spring Security LDAP
  - Spring Security OpenID
  - Spring Security CAS
  - Spring Security Cryptography
  - Spring Security Remoting

Resumen de la lección

vii.i Introducción

- Comprendimos qué es Spring Security.
- Comprendimos las funcionalidades que provee Spring Security.
- Analizamos las ventajas que tiene Spring Security frente a otros frameworks de seguridad.
- Conocimos a grandes rasgos los módulos y sub-módulos de los proyectos asociados a Spring Security.

Esta página fue intencionalmente dejada en blanco.

vii.i Spring Security Web MVC

Objetivos de la lección

vii.ii Spring Security Web MVC

- Analizar algunas de los problemas de seguridad más comunes en aplicaciones web.
- Comprender a grandes rasgos cómo Spring Security implementa seguridad en aplicaciones web.
- Conocer las librerías relacionadas a los módulos y sub-módulos de Spring Security.
- Conocer las libarías asociadas a la implementación de Spring Security en contexto de Spring MVC.

vii.ii Spring Security Web MVC

a. Dependencias

vii.ii Spring Security Web MVC (a)

- Spring Security implementa seguridad en aplicaciones Java EE de diferentes contextos.

- Spring Security se integra a Spring MVC así como a Spring Web (basado en Servlets)

- Las aplicaciones web requieren seguridad, en la mayoría de los casos, los ataques a un servidor web se dan mediante un el mal uso de la aplicación web que accede a sus recursos.

vii.ii Spring Security Web MVC (b)

- La seguridad de aplicaciones en contexto Web es crítico, por tanto es requerido implementar soluciones probadas y no “reinventar la rueda”.

- Spring Security Web MVC previene de ataques CSRF así como *Session Fixation*.

- Spring Security Web MVC se integra fácilmente a aplicaciones Spring MVC, de forma no intrusiva mediante filtros.

vii.ii Spring Security Web MVC (c)

- Provee diferentes maneras (implementaciones) de autenticación.
- Soporta opción para ignorar patrones URL específicos tal como contenido estático.
- Soporta autorización mediante roles.

vii.ii Spring Security Web MVC

a. Dependencias

vii.ii Spring Security Web MVC (a)

- Dependencias.

- Para integrar Spring Security a cualquier aplicación es necesario agregar las dependencias correspondientes.
  - spring-security-core-<version>.jar
  - spring-security-config-<version>.jar
  - spring-security-web-<version>.jar
  - spring-security-taglibs-<version>.jar
  - spring-security-aspects-<version>.jar
  - spring-security-acl-<version>.jar
  - spring-security-ldap-<version>.jar
  - spring-security-openid-<version>.jar
  - spring-security-cas-<version>.jar
  - spring-security-crypto-<version>.jar
  - spring-security-remoting-<version>.jar

vii.ii Spring Security Web MVC (b)

- Dependencias.

- Para la implementación de Spring Security en una aplicación Spring MVC será necesario agregar las siguientes dependencias:
  - spring-security-core-<version>.jar
  - spring-security-config-<version>.jar
  - spring-security-web-<version>.jar
  - spring-security-taglibs-<version>.jar
  - spring-security-aspects-<version>.jar
  - spring-security-acl-<version>.jar
  - spring-security-ldap-<version>.jar
  - spring-security-openid-<version>.jar
  - spring-security-cas-<version>.jar
  - spring-security-crypto-<version>.jar
  - spring-security-remoting-<version>.jar

vii.ii Spring Security Web MVC (c)

- Dependencias.

- Maven dependency: El groupId es diferente de Spring Frameowrk (Core). La versión de Spring Framework y Spring Security no necesariamente debe ser la misma.

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>Module name</artifactId>
    <version>${spring-security-version}</version>
</dependency>
```

Resumen de la lección

vii.ii Spring Security Web MVC

- Analizamos los problemas de Seguridad CSRF así como Session Fixation.
- Comprendimos que Spring Security se implementa a través de Filtros del API Servlet.
- Conocimos las librerías que comprenden el proyecto Spring Security.
- Comprendimos la forma de agregar las dependencias requeridas de Spring Security en contexto web con Spring MVC.

Esta página fue intencionalmente dejada en blanco.

vii.iii Implementación de capa de Seguridad Web

Objetivos de la lección

vii.iii Implementación de capa de Seguridad Web (a)

- Conocer los principales objetos relacionados con la seguridad que implementa Spring Security.
- Comprender la forma de configurar Spring Security mediante el archivo descriptor de la aplicación web (web.xml).
- Analizar el namespace <security>
- Comprender la definición de filtros (interceptores) para implementar seguridad a nivel de URL.
- Implementar seguridad a nivel de método de clase.

Objetivos de la lección

vii.iii Implementación de capa de Seguridad Web (b)

- Conocer la implementación de formulario de login.
- Implementar la configuración de login, logout y acceso denegado.

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml
b. Configuración Spring Security Context
c. Formulario Login / Logout
d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web (a)

- Para comprender como se implementa la seguridad en Spring Security es necesario revisar algunos conceptos:
  - Authentication
  - Authorization
  - UserDetails (principal)
  - GranthedAuthority
  - UserDetailsService
  - AuthenticationProvider
  - AuthenticationManager
  - SecurityContext
  - SecurityContextHolder

vii.iii Implementación de capa de Seguridad Web (b)

- Authentication:
  - Representa la autenticación del usuario, contiene el objeto *principal* (instancia de la clase User, impl. de UserDetails).

```
public interface Authentication extends Principal, Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();
    Object getCredentials();
    Object getDetails();
    Object getPrincipal();
    boolean isAuthenticated();
    void setAuthenticated(boolean isAuthenticated) throws IllegalArgumentException;
}
```

vii.iii Implementación de capa de Seguridad Web (c)

- Authorization:
  - Representa los permisos otorgados al objeto *principal*.
  - Los permisos se representan mediante la interface GrantedAuthority.

  - La autenticación, como verbo, hace referencia a la validación de usuario y contraseña.
  - La autorización, como sustantivo, es el nivel de acceso o permisos otorgados al objeto autenticado.

vii.iii Implementación de capa de Seguridad Web (d)

- UserDetails:
  - Interface que provee de la información necesaria del objeto *principal* (clase User implementación por defecto).

```
public interface UserDetails extends Serializable {
    Collection<? extends GrantedAuthority> getAuthorities();
    String getPassword();
    String getUsername();
    boolean isAccountNonExpired();
    boolean isAccountNonLocked();
    boolean isCredentialsNonExpired();
    boolean isEnabled();
}
```

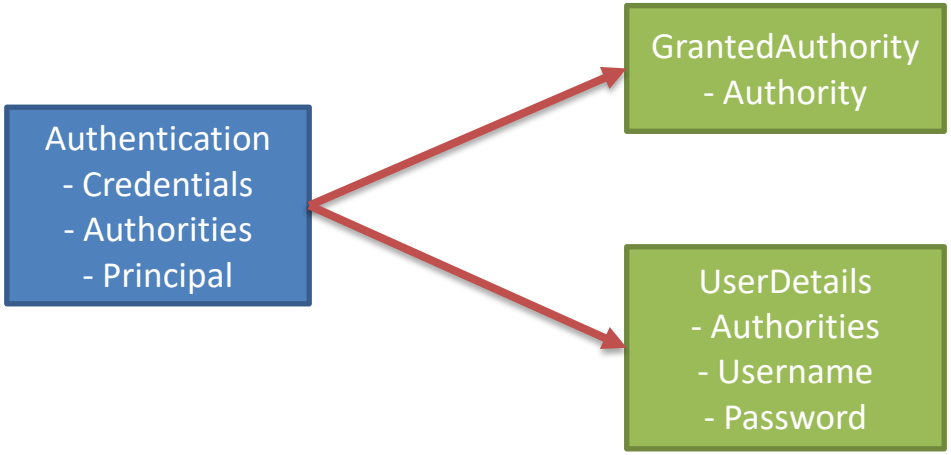
vii.iii Implementación de capa de Seguridad Web (e)

- GrantedAuthority:
  - Interface que provee la definición del nivel de acceso o permisos del objeto *principal* autenticado.

```
public interface GrantedAuthority extends Serializable {
    String getAuthority();
}
```

vii.iii Implementación de capa de Seguridad Web (f)

- Authentication, GrantedAuthority y UserDetails



vii.iii Implementación de capa de Seguridad Web (g)

- UserDetailsService:
  - Interface que provee el mecanismo para obtener, desde cualquier origen de datos (por ejemplo un DAO), la información del usuario a autenticar mediante su *username*.

```
public interface UserDetailsService {
    UserDetails loadUserByUsername(String username) throws
                                                    UsernameNotFoundException;
}
```

vii.iii Implementación de capa de Seguridad Web (h)

- AuthenticationProvider:
  - Provee el mecanismo por el cual el *AuthenticationManager* autenticará usuarios. Este proveedor puede contener múltiples servicios (UserDetailsService) para delegar la autenticación hacia distintos orígenes de datos.

```
public interface AuthenticationProvider {
    Authentication authenticate(Authentication authentication) throws
                                                        AuthenticationException;
    boolean supports(Class<?> authentication);
}
```

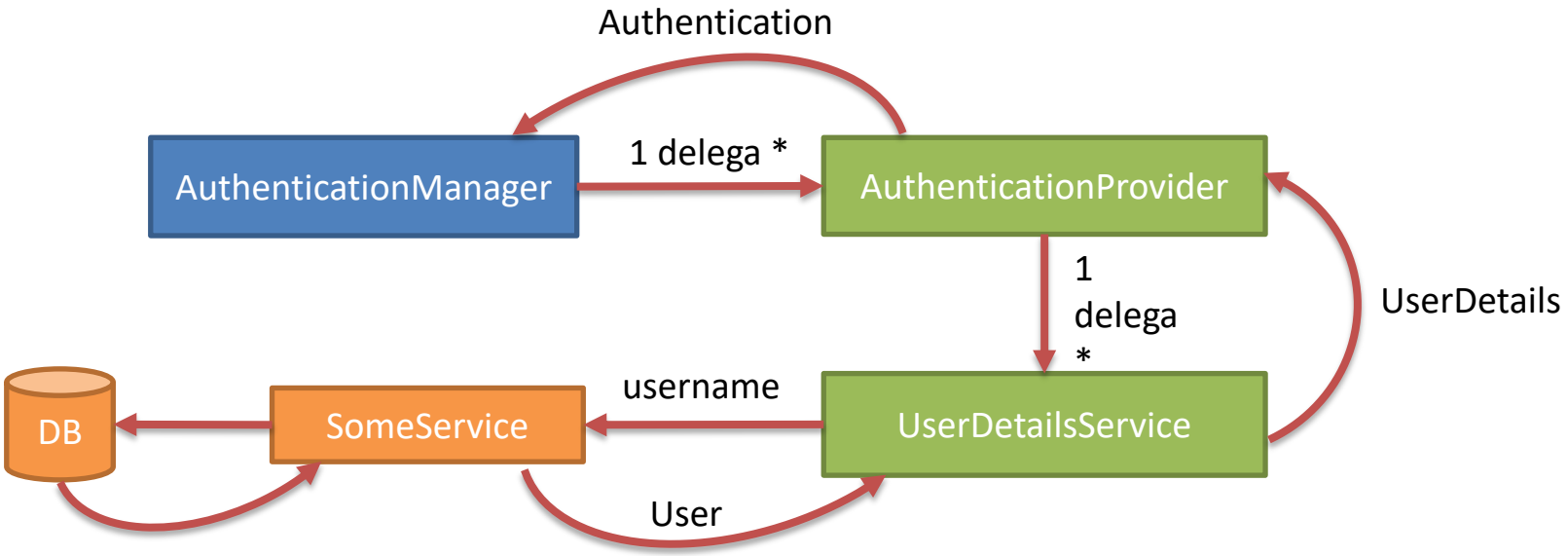
vii.iii Implementación de capa de Seguridad Web (i)

- AuthenticationManager:
  - Es el objeto que se encarga de manejar las peticiones de autenticación desde cualquier parte de la aplicación (es un bean manejador). En el es posible registrar distintos *AuthenticationProvider.*

```
public interface AuthenticationManager {
    Authentication authenticate(Authentication authentication) throws
                                                        AuthenticationException;
}
```

vii.iii Implementación de capa de Seguridad Web (j)

- UserDetailsService , AuthenticationProvider y AuthenticationManager



vii.iii Implementación de capa de Seguridad Web (k)

- SecurityContext:
  - Mantiene y permite acceder la referencia del objeto *Authentication.*

```
public interface SecurityContext extends Serializable {
    Authentication getAuthentication();
    void setAuthentication(Authentication authentication);
}
```

vii.iii Implementación de capa de Seguridad Web (I)

- SecurityContextHolder:
  - Provee acceso al objeto *SecurityContext* mediante la implementación de una estrategia (*SecurityContextHolderStrategy*), comúnmente usa la clase ThreadLocal para almacenar los detalles del *SecurityContext,* y así poder acceder a sus detalles mediante un único hilo.

```
public class SecurityContextHolder {
    …
    public static SecurityContext getContext() {…}
    …
}
```

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml

b. Configuración Spring Security Context

c. Formulario Login / Logout

d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web (a)

- Configuración web.xml

- La infraestructura web para la implementación de Spring Security se basa en el estándar API Servlet mediante la implementación de filtros.

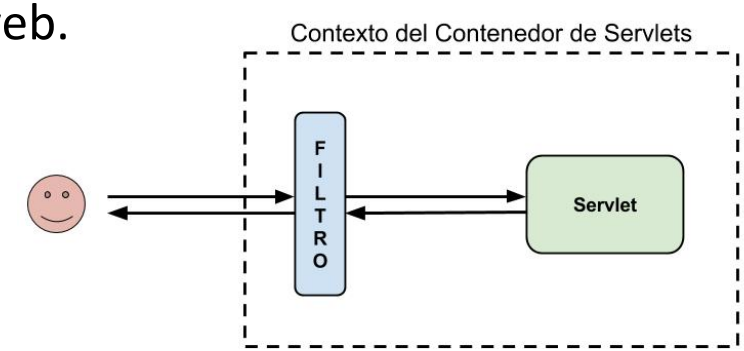
- Esta fuertemente desacoplado de cualquier framework o tecnología web.

- Mediante filtros, Spring Security, manipula la petición HttpServletRequest y la respuesta HttpServletResponse sin importar si la petición proviene de un cliente web, una petición AJAX o mediante HttpInvoker.

vii.iii Implementación de capa de Seguridad Web (b)

- Configuración web.xml

- Spring Security se configura en base a una cadena de filtros (SpringSecurityFilterChain), el cual se encarga de orquestar las distintas validaciones que se deben de ejecutar para lograr la seguridad en la aplicación web.



vii.iii Implementación de capa de Seguridad Web (c)

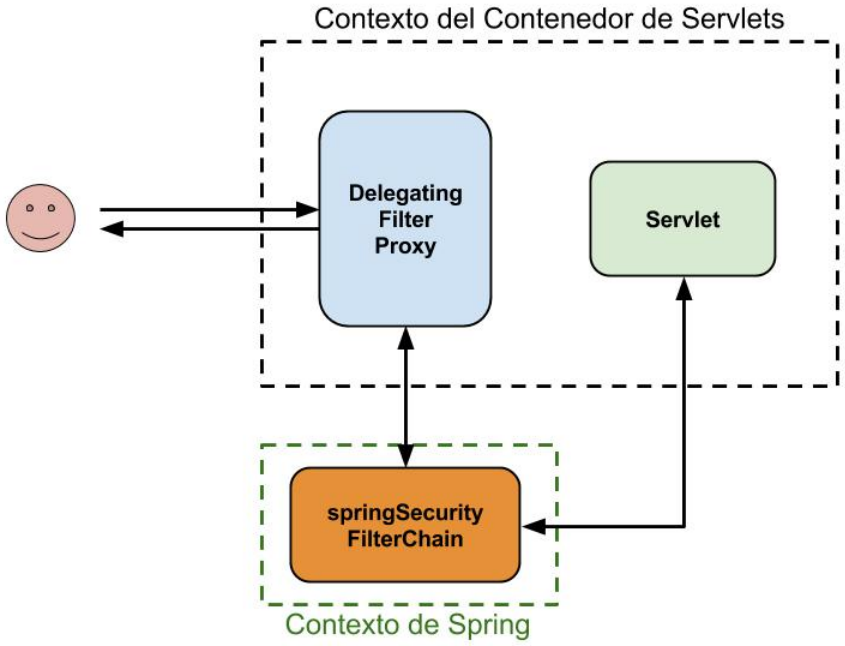
- Configuración web.xml

- El filtro DelegatingFilterProxy se encarga de buscar, en el contexto de Spring, un bean llamado *springSecurityFilterChain* en el cual delegará las comprobaciones de seguridad relativas a la aplicación web.

```
<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>
```

vii.iii Implementación de capa de Seguridad Web (d)

- Configuración web.xml: DelegatingFilterProxy



vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml

b. Configuración Spring Security Context

c. Formulario Login / Logout

d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web (a)

- Configuración Spring Security Context

- El contexto de beans de Spring Security debe ser registrado en el *RootApplicationContext* implementación de *ApplicationContext*, no a nivel del *WebApplicationContext*.

- Spring Security no es parte de Spring MVC, es una implementación de beans distinta y sus beans deben ser manejados por el contenedor de IoC de Spring por defecto (ApplicationContext).

vii.iii Implementación de capa de Seguridad Web (b)

- Configuración Spring Security Context

- Por tanto, para registrar el contexto de Spring Security lo realizamos mediante *ContextLoaderListener*.

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-application-context.xml,
                 /WEB-INF/spring/security/spring-security-application-context.xml</param-value>
</context-param>

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

vii.iii Implementación de capa de Seguridad Web (c)

- Configuración Spring Security Context

- Una vez inicializada la carga del contexto de Spring Security en el *RootApplicationContext* es necesario definir el contexto de seguridad mediante el namespace <security>

```
<beans xmlns="http://www.springframework.org/schema/beans" …
    xmlns:security="http://www.springframework.org/schema/security"
    xsi:schemaLocation="http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd>
</beans>
```

vii.iii Implementación de capa de Seguridad Web (d)

- Configuración infraestructura básica Spring Security

```
<security:authentication-manager>
    <security:authentication-provider>
        <security:user-service>
            <security:user name="admin" password="admin“ authorities="ROLE_ADMIN" />
            <security:user name="xvanhalenx" password="123123“
                                                    authorities="ROLE_ROOT,ROLE_ADMIN" />
            <security:user name="user" password="user“ authorities="ROLE_USER" />
        </security:user-service>
    </security:authentication-provider>
</security:authentication-manager>
```

In Memory UserService Authentication

vii.iii Implementación de capa de Seguridad Web (e)

- Configuración de autorización a nivel de URL

```
<security:http auto-config="true" use-expressions="true“ … >
    …
    <security:intercept-url pattern="/welcome“ access="isAuthenticated()" />
    <security:intercept-url pattern="/login“ access="permitAll" />
    <security:intercept-url pattern="/root*/**“
                                                    access="hasAnyRole('ROLE_ROOT','ROLE_ADMIN')" />
    <security:intercept-url pattern="/admin*/**“ access="hasAnyRole('ROLE_ADMIN')" />
    <security:intercept-url pattern="/user*/**“ access="hasRole('ROLE_USER')" />

</security:http>
```

vii.iii Implementación de capa de Seguridad Web (f)

- Configuración de URIs de login/logout

```
<security:http auto-config="true" use-expressions="true“ access-denied-page="/access-denied">

    <security:form-login login-page="/login“ authentication-failure-url="/login?error=true"
                                                                    default-target-url="/welcome" />

    <security:logout invalidate-session="true“ logout-success-url="/login" logout-url="/logout" />
    …

</security:http>
```

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml

b. Configuración Spring Security Context

c. Formulario Login / Logout

d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web (a)

- Formulario Login/Logout

- Cuando se configura Spring Security con Spring MVC, es posible implementar un formulario de forma automática, sin necesidad de desarrollar como tal un formulario, sin embargo no es personalizable (estilos css, funcionalidad js).

- Una vez definido el URI que apunta a la página de *login (<security:form-login login-page="/login“ ../>)* es necesario crear un controlador que intercepte esa URI y defina un formulario JSP.

vii.iii Implementación de capa de Seguridad Web (b)

- Formulario Login

```
@RequestMapping(value = { "", "/", "/login" }, method = RequestMethod.GET)
public String showLoginPage(Model model, @RequestParam(required = false, value = "error")
                                                                    boolean error) {
    if (error) {
        model.addAttribute("errorMessage", "Wrong user or password");
    }
    return "login/view_login";
}
```

vii.iii Implementación de capa de Seguridad Web (c)

- Formulario Login

```
<security:authorize access="isAnonymous()">
    <div id="login-error" class="error">${errorMessage}</div>
    <form id="form" action="j_spring_security_check" method="post">
        Username: <input id="j_username“ name="j_username" type="text" /> <br />
        Password: <input id="j_password" name="j_password" type="password" /> <br />

        <input class="button" type="submit" value="Login" />
    </form>
</security:authorize>

...
```

vii.iii Implementación de capa de Seguridad Web (d)

- Formulario Login

```
...

<security:authorize access="isAuthenticated()">
    <h2>¡ User '<security:authentication property="principal.username" />' you're
        already authenticated !</h2>
    <security:authentication property="principal.authorities" />
</security:authorize>
```

vii.iii Implementación de capa de Seguridad Web (e)

- Formulario Login

- De existir un error en autenticación *(login)*, se invocará a la URI */login?error=true*

- De ser valida la comprobación del usuario y contraseña *(login)*, se invocará a la URI */welcome* (es necesario definir el controlador y su correspondiente vista).

vii.iii Implementación de capa de Seguridad Web (f)

- Formulario Logout

- Para realizar *logout*, es necesario hacer una invocación a la URI */logout* (no es necesario implementar el controlador, lo gestiona Spring Security en automático).

- Si el *logout* se ejecutó satisfactoriamente se invocará a la URI */login* nuevamente.

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml
b. Configuración Spring Security Context
c. Formulario Login / Logout
d. Global Method Security
Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web (a)

- Global Method Security

- Es posible aplicar seguridad mediante expresiones a métodos, ya sea métodos *handler method* de controlador o métodos de servicio, siempre y cuando el método a asegurar sea miembro de un bean de Spring.

- Para implementar seguridad a métodos de beans primeramente es necesario habilitar:

```
<security:global-method-security pre-post-annotations="enabled" />
o
@EnableGlobalMethodSecurity(prePostEnabled = true)
```

vii.iii Implementación de capa de Seguridad Web (b)

- Global Method Security

- Para asegurar el acceso a nivel de método es necesario utilizar las anotaciones @PreAuthorize y @PostAuthorize.

- @PreAuthorize: define si el método será ejecutado o no (recomendado).

- @PostAuthorize: define si se devolverá o no, el método ejecutado (no recomendado).

- Global Method Security requiere AOP para funcionar.

vii.iii Implementación de capa de Seguridad Web (c)

- Global Method Security

- Es necesario agregar al classpath las dependencias:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-aspects</artifactId>
</dependency>
```

vii.iii Implementación de capa de Seguridad Web

a. Configuración web.xml
b. Configuración Spring Security Context
c. Formulario Login / Logout
d. Global Method Security

Práctica 32. Implementación Spring Security Web MVC

vii.iii Implementación de capa de Seguridad Web.
Práctica 32. (a)

- Práctica 32. Implementación Spring Security Web MVC

- Configurar Spring Security en un contexto web *(WebApplicationContext)*.
- Implementar seguridad a nivel de URL y a nivel de método.
- Implementar formulario de acceso a aplicaciones web.
- Configurar los distintos beans necesarios para implementar Spring Security.

Resumen de la lección

vii.iii Implementación de capa de Seguridad Web (a)

- Comprendimos la utilidad de las principales interfaces que gestionan el manejo de autenticación y autorización en Spring Security.
- Comprendimos como se relacionan las interfaces principales de Spring Security.
- Revisamos como implementar el Filtro de Spring Security.
- Configuramos los beans Requeridos de Spring Security.
- Implementamos un formulario de login así como la implementación de logout.

Resumen de la lección

vii.iii Implementación de capa de Seguridad Web (b)

- Verificamos como implementar seguridad a nivel de URL.
- Analizamos como implementar seguridad a nivel de método.

Esta página fue intencionalmente dejada en blanco.