

➤ Control de Acceso mediante Spring Security

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Implementar y configurar la transaccionalidad en los servicios
- ✓ Crear servicios transaccionales

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

6.3

Start! 🏁

**Control de Acceso
mediante Spring
Security**

Spring Security

Módulo de seguridad

Incorporando el módulo Spring Security al proyecto
Configuración básica de Spring Security

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Aprender a incorporar el módulo Spring Security al proyecto



Comprender la configuración básica de Spring Security



› Spring Security



Spring Security



Spring Security es un marco de autenticación y control de acceso potente y altamente personalizable. Es el estándar de facto para proteger las aplicaciones basadas en Spring. Es un marco que se enfoca en proporcionar **autenticación** y **autorización** a las aplicaciones Java.





Spring Security



Características principales:

1- Autenticación y Autorización:

- **Proveedores de Autenticación:** Spring Security admite múltiples proveedores de **autenticación**, como autenticación basada en formularios, autenticación basada en tokens (por ejemplo, JWT), autenticación LDAP, etc.
- **Roles y Permisos:** La **autorización** se basa en roles y permisos. Los roles agrupan permisos y se asignan a los usuarios para controlar su acceso a recursos específicos.





Spring Security



Características principales:

2- Configuración Declarativa:



- **Anotaciones:** Spring Security permite la configuración declarativa mediante el uso de anotaciones, como `@Secured`, `@PreAuthorize`, y `@PostAuthorize`, que se aplican a nivel de método.
- **Configuración Java:** También se puede realizar una configuración declarativa en Java mediante clases que extienden `WebSecurityConfigurerAdapter`.





Spring Security



Características principales:

3- Protección de Recursos:



- **Configuración de Reglas de Acceso:** Se pueden definir reglas de acceso para proteger recursos específicos. Esto incluye restricciones basadas en **URL**, métodos **HTTP** y reglas personalizadas.
- **Control de Sesiones:** Spring Security proporciona opciones para gestionar **sesiones**, incluyendo la posibilidad de limitar el número de sesiones por usuario.





Spring Security



Características principales:

4- Gestión de Sesiones y Tokens:



- **Control de Sesiones:** Spring Security ofrece mecanismos para gestionar la creación y finalización de sesiones, así como la detección de sesiones inactivas.
- **Autenticación Basada en Tokens:** Puede implementar la autenticación basada en tokens, como **JSON Web Tokens** (JWT), para permitir la autenticación sin el uso de sesiones.





Spring Security



Características principales:



5- Integración con Spring Boot:



- **Autoconfiguración:** Spring Security se integra sin problemas con Spring Boot y ofrece una configuración automática basada en **convenciones**.
- **Propiedades de Aplicación:** Puede personalizar la configuración de seguridad mediante propiedades en el archivo de configuración de la aplicación (application.properties o application.yml).





Spring Security



Características principales:

6- Manejo de Errores y Eventos:



- **Manejo de Errores:** Spring Security proporciona mecanismos para personalizar las páginas de error de autenticación y acceso denegado.
- **Eventos del Ciclo de Vida:** Puede aprovechar los eventos del ciclo de vida de Spring Security para ejecutar lógica personalizada en diversas etapas, como autenticación exitosa, autenticación fallida, etc.





Spring Security



Características principales:



7- Protección contra Ataques Comunes:



- **Protección contra CSRF:** Spring Security incluye medidas para prevenir ataques de falsificación de solicitudes entre sitios (**CSRF**).
- **Protección contra Inyecciones:** Proporciona protección contra ataques de inyección, como ataques de inyección SQL.





Spring Security



Características principales:

8- Auditoría y Registro:



- **Auditoría de Accesos:** Permite la auditoría de eventos de seguridad, como accesos exitosos o fallidos, mediante configuración o eventos.
- **Registros Detallados:** Proporciona registros detallados de eventos de seguridad, lo que facilita la detección y resolución de problemas de seguridad.





Spring Security



Características principales:

9- Extensibilidad y Personalización:

- **Puntos de Extensión:** Spring Security ofrece numerosos puntos de extensión que permiten personalizar su comportamiento, como proveedores de autenticación personalizados, manejadores de acceso denegado personalizados, etc.
- **Personalización de Flujo de Autenticación:** Puede personalizar el flujo de autenticación para adaptarse a los requisitos específicos de la aplicación.





Spring Security



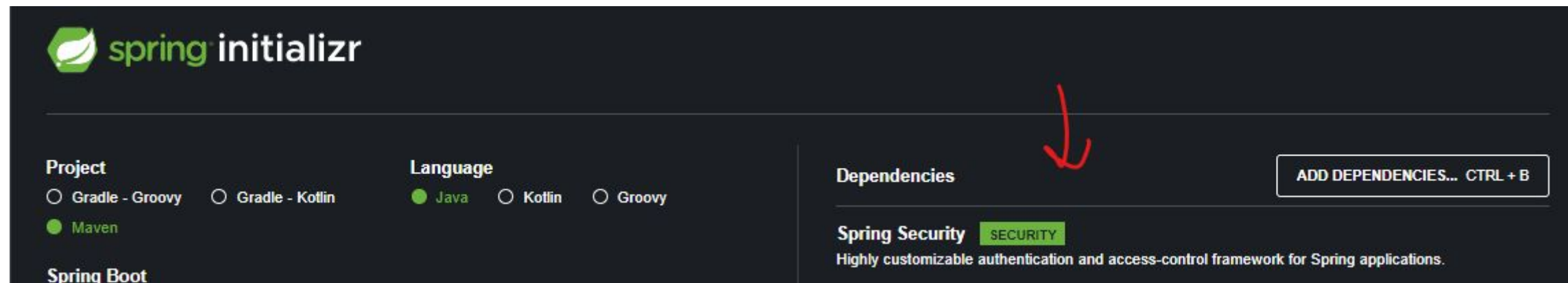
La seguridad es un componente crítico en cualquier aplicación web moderna. Spring Security es una poderosa extensión de Spring que proporciona un marco integral para la autenticación y autorización, ayudando a proteger nuestras aplicaciones contra amenazas y garantizando que solo los usuarios autorizados accedan a los recursos adecuados.



➤ Incorporando Spring Security al proyecto

Incorporando Spring Security al proyecto

Añadir Spring Security en cualquier proyecto Spring de Java es muy sencillo, sobre todo si se ha utilizado la capa de auto-configuración ya que el uso de Spring Boot simplifica enormemente las acciones de configuración que debe realizar el desarrollador para poner en marcha cualquier aplicación Spring.



Incorporando Spring Security al proyecto

En caso de decidir agregar Spring Security en un proyecto ya creado y configurado se debe agregar la dependencia correspondiente en el archivo **pom.xml**.

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

Incorporando Spring Security al proyecto

La configuración básica de Spring Security se puede establecer en el archivo de configuración de la aplicación. Por ejemplo, en `application.properties`:

```
# Configuración básica de Spring Security
spring.security.user.name=user
spring.security.user.password=password
spring.security.user.roles=USER
```

Esto establece un usuario predeterminado con nombre 'user', contraseña 'password' y rol 'USER'



Incorporando Spring Security al proyecto

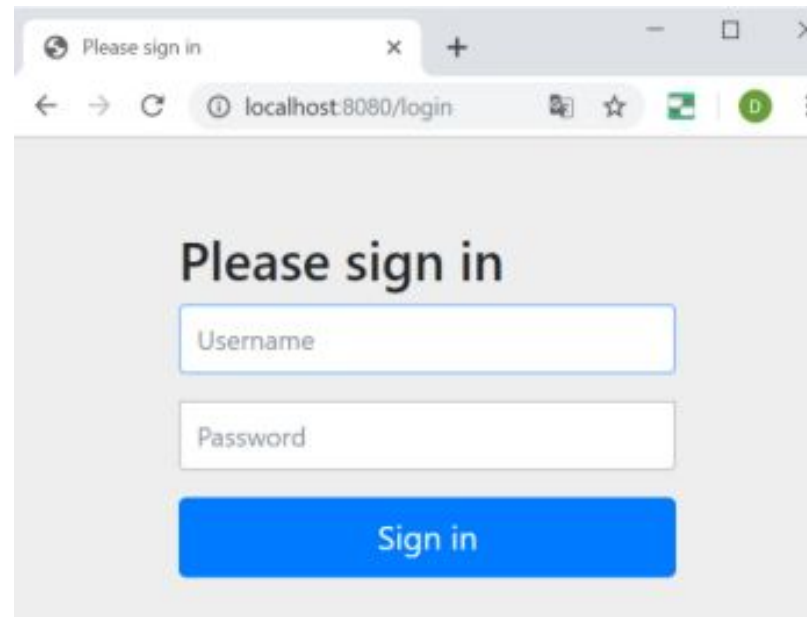


Por **sólo incluir la dependencia**, si el programador no realiza ninguna configuración adicional, Spring Boot de manera predeterminada protegerá todo el acceso a la aplicación, impidiendo que ningún usuario no identificado pueda invocar a cualquier controlador. Este mecanismo tan restrictivo suele ser suficiente para pequeñas aplicaciones donde sólo se necesita restringir el acceso de forma general, pero queda algo reducido en aplicaciones donde hay secciones accesibles y otras protegidas, dependiendo de si el usuario está identificado o de si éste tiene un rol determinado.



Incorporando Spring Security al proyecto

Con ésta configuración, si creamos un endpoint cualquiera e intentamos consumirlo, se mostrará un formulario de inicio de sesión proporcionado por Spring Security:



The screenshot shows a web browser window with the title "Please sign in". The address bar displays "localhost:8080/login". The page content includes the heading "Please sign in", a text input field labeled "Username", another text input field labeled "Password", and a blue button labeled "Sign in".

Incorporando Spring Security al proyecto

Si compruebas la salida por consola, se podrá ver una contraseña generada automáticamente. El nombre de usuario por defecto es «user».

```
Using generated security password: 30f20aec-fa73-4dee-a972-22f8ff77a1a5
```


Incorporando Spring Security al proyecto

Para poder realizar la configuración, debemos crear una clase que extienda de `WebSecurityConfigurerAdapter` y tiene que estar anotada como `@Configuration` y `@EnableWebSecurity`.

```
1 @Configuration
2 @EnableWebSecurity
3 public class SeguridadWeb extends WebSecurityConfigurerAdapter {
```

Incorporando Spring Security al proyecto

Lo que haremos será sobrescribir el método `configure` para poder configurar nosotros el **AuthenticationManagerBuilder** que es el objeto que se encargará de realizar el manejo de la autenticación de usuarios.

- **Autenticación:** verificamos la identidad del usuario.
- **Autorización:** tipo de permisos que tiene ese usuario

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
@EnableWebMvc
public class SeguridadWeb extends WebSecurityConfigurerAdapter {

    @Autowired
    public UserService usuarioServicio;

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception{
        auth.userDetailsService(usuarioServicio);
    }
}
```

Incorporando Spring Security al proyecto

Para una configuración más avanzada, podemos sobrescribir el método **configure** para personalizar los accesos de la aplicación:

```
@Configuration
@EnableWebSecurity
public class SeguridadWeb extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests()
            .antMatchers("/public/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin(login -> login
                .loginPage("/login")
                .permitAll())
            .logout(logout -> logout
                .permitAll());
    }
}
```



Incorporando Spring Security al proyecto



En este ejemplo, configuramos reglas de autorización y autenticación.

Permitimos el acceso a todas las URL que coincidan con **"/public/**"**, requerimos autenticación para cualquier otra URL, configuramos una página de inicio de sesión personalizada, y permitimos que todos los usuarios cierren sesión.



```
@Configuration
@EnableWebSecurity
public class SeguridadWeb extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests()
            .antMatchers("/public/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin(login -> login
                .loginPage("/login")
                .permitAll()
            )
            .logout(logout -> logout
                .permitAll());
    }
}
```

Momento: ✚

Time-out!

🕒 5 min.



Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.



LIVE CODING

Ejemplo en vivo

Seguridad en la Wallet

- *Vamos a agregar la dependencia de Spring Security en el proyecto AlkeWallet*
- *Además, agregaremos las configuraciones necesarias en el archivo **application.properties***
- *También crearemos una nueva clase: **SecurityConfig**, que debe extender de `WebSecurityConfigurerAdapter` y debe sobrescribir el método `configure` para personalizarlo.*

 **Tiempo: 25 minutos**



Ejercicio

Módulo de seguridad



Módulo de seguridad



Contexto: 🙌

Vamos a continuar con el ejercicio de la clase anterior. En este caso vamos a agregar una el módulo de seguridad de Spring Security,



Consigna: 📝

- 1- Buscar la dependencia Maven para Spring Security e integrar al proyecto (pom.xml)
- 2- Correr el proyecto e ingresar los datos de inicio de sesión que configura Spring de manera automática.

Tiempo 🕒: 15 minutos

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender el módulo de seguridad de Spring Security**
- ✓ **Aprender a configurar Spring Security en un proyecto**



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 📌 📌 📌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. *Lectura Modulo 6, Lección 4: páginas 1 - 5*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

Momento: ✚

Time-out!

🕒 5 min.

