

SECURIZACIÓN DE MICROSERVICIOS



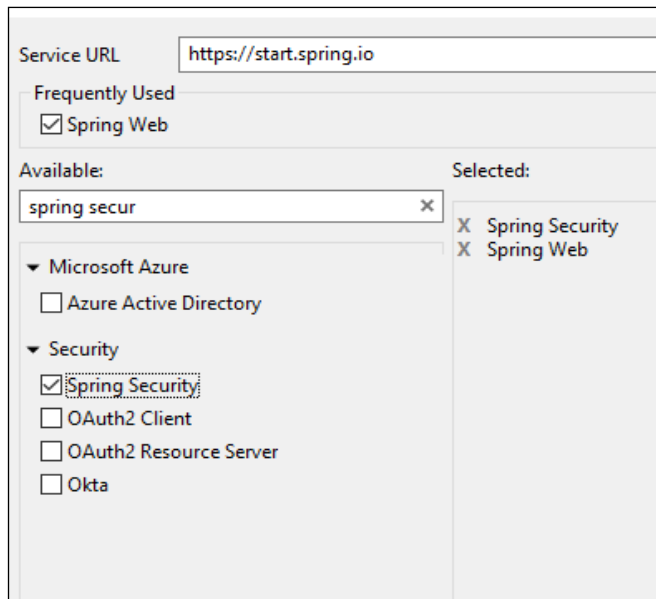
Concepto

- **Securizar un servicio consiste en permitir el acceso a los recursos, solamente a usuarios autorizados**
- **Se requiere un proceso de autenticación y autorización**
- **Gestionado por Spring Security**



Dependencias

➤ Se debe incorporar el starter de spring security



The screenshot shows the Spring Boot Start web interface. At the top, the 'Service URL' is set to 'https://start.spring.io'. Under the 'Frequently Used' section, 'Spring Web' is checked. In the 'Available:' section, a search bar contains 'spring secur' and 'Spring Security' is selected. Below the search bar, there are two expandable sections: 'Microsoft Azure' and 'Security'. Under 'Security', 'Spring Security' is checked, while 'OAuth2 Client', 'OAuth2 Resource Server', and 'Okta' are unchecked. To the right of the 'Available:' section, the 'Selected:' section shows 'Spring Security' and 'Spring Web' with an 'X' next to each, indicating they are added to the project dependencies.

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



Configuración

¡deprecated desde
Spring boot 2.7!

➤ Se establece en una clase de seguridad que extiende **WebSecurityConfigurerAdapter**:

Definición de
roles y usuarios

```
@EnableWebSecurity
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter{
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        ..
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        ..
    }
}
```

Políticas de acceso
a recursos



Configuración

A partir de Spring
boot 2.7

➤ Se definen dos métodos para creación de objetos:

Definición de
roles y usuarios

```
@EnableWebSecurity
@Configuration
public class SecurityConfig {
    @Bean
    public InMemoryUserDetailsManager usersDetails () throws Exception {
        List<UserDetails> users=List.of(
            User.withUsername("user1")
                .password("{noop}user1")
                .roles("USER")
                .build(),
            ....
        );
        return new InMemoryUserDetailsManager(users);
    }
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        ..
        return http.build();
    }
}
```

Políticas de acceso
a recursos

