Introduction to Spring Security and its Features

Last Updated : 23 Jul, 2025

Spring Security is a powerful **authentication and authorization** framework used to **secure Java-based web applications**. It easily integrates with Spring Boot and provides advanced security mechanisms such as **OAuth2, JWT-based authentication, role-based access control**, and protection against common vulnerabilities like **CSRF, session fixation,** and **brute-force attacks**. With the latest enhancements in **Spring Security 6.4**, developers can apply modern security configurations, reactive programming support, and improved password encoding strategies. This article provides an in-depth overview of Spring Security, its features, and its implementation using Maven dependencies.

Core Concepts of Spring Security

Spring Security is built on four core concepts:

- **Authentication**: Verifies the user's identity.
- **Authorization**: Determines user permissions and access control.
- **Password Storage**: Securely manages password encoding and storage.
- **Servlet Filters**: Manages security filters to control requests.

Advantages of Spring Security

Some major benefits of using Spring Security include:

- Protection against threats like CSRF, session fixation, and clickjacking.
- Integration with Spring MVC and Spring Boot.
- Supports Java-based configuration.
- Works with standard Servlet API.
- Prevents brute-force attacks.

- Active open-source community ensuring continuous improvements.

Maven Setup for Spring Security

Setting up spring-security-core

Add the following dependency to your pom.xml:

```
<properties>
      <spring-security.version>6.4.0</spring-security.version>
      <spring.version>6.1.0</spring.version>
</properties>


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

Setting up spring-security-web

Add the following dependency:

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

Using **spring-boot-starter-security** is the recommended approach, as it ensures Maven pulls the correct version of Spring Security based on your Spring Boot version, reducing dependency conflicts.

Spring Security Features

Spring Security provides various security features, including:

1. **Authorization**: Controls access based on user roles and permissions.

2. **Single Sign-On (SSO)**: Enables users to log in once and access multiple applications.
3. **Software Localization**: Supports multi-language authentication interfaces.
4. **Remember-Me**: Allows persistent user sessions using cookies.
5. **LDAP Support:** Integrates with Lightweight Directory Access Protocol (LDAP) for authentication.
6. **JAAS Integration**: Supports Java Authentication and Authorization Service (JAAS).
7. **Web Form Authentication**: Handles user credentials through web-based forms.
8. **Digest Access Authentication**: Provides enhanced security over basic authentication.
9. **HTTP Authorization**: Uses patterns or regex to manage URL-based authorization.
10. **Basic Access Authentication**: Implements basic authentication for network requests.

Latest Features in Spring Security 6.4 (2025 Update)

Spring Security 6.4 introduces several new improvements:

- **OAuth 2.0 Enhancements**: Allows easy integration with third-party login providers like Google, GitHub, and Microsoft.
- **Improved Reactive Security:** Enhanced support for Spring WebFlux and non-blocking security handling.
- **Advanced Password Encoding**: Utilizes **DelegatingPasswordEncoder** with multiple encoding formats:

  {bcrypt}$2a$10$...hashed-password...

  {noop}password

  {pbkdf2}hashed-password...

  {scrypt}hashed-password...

  {sha256}hashed-password...

- **Unified Request Matchers API:** Methods like antMatchers(), mvcMatchers(), and regexMatchers() are deprecated in favor of requestMatchers().
- **Enhanced Authorization Rules**: More flexible security configurations for role-based and method-level security.
- **Security Context Propagation**: Improved support for security context sharing across different execution models, including reactive and asynchronous flows.

Note: Always check the official Spring documentation for the latest updates, security patches, and version changes: Spring Security Official Documentation