


Computer science → Backend → Spring Boot → Spring Security

# Getting started with Spring Security

Theory

Practice

 10% completed, 0 problems solved

▼

## Theory

🕒 9 minutes reading

Verify to skip

Start practicing

Access to some web pages, files, or other classified resources of a web application is often restricted to authorized users only. **Spring Security** is a module of the Spring framework that deals with authentication and authorization (or access control). It stands between the client and the application, intercepts all requests, and allows configuring what functionalities and data are available to which users. It also helps secure your app against common security vulnerabilities and attacks.

Some of the features of this framework are the following:

- Easily configurable and extensible to meet the needs of a specific application.
- Protection against common attacks: session fixation, clickjacking, CSRF, and so on.
- Open source and regular updates.
- Supports integration with HTTP basic access authentication, form-based authentication, LDAP, and many more.
- Provides a secure and flexible set of tools for managing user passwords.

In this topic, you'll learn how to start working with Spring Security and we'll create a secured single-page app.

## §1. Dependency

The first step in securing a Spring app is to add the Spring Boot Security starter dependency:

### ▼ Gradle Groovy DSL

```
1 dependencies {
2     // ...
3     implementation 'org.springframework.boot:spring-boot-starter-
security'
4     // ...
5 }
```

### ▼ Gradle Kotlin DSL


```
1 dependencies {
2     // ...
3     implementation("org.springframework.boot:spring-boot-starter-
security")
4     // ...
5 }
```

### ▼ Maven

```
1 <dependency>
2     <groupId>org.springframework.boot</groupId>
3     <artifactId>spring-boot-starter-security</artifactId>
4 </dependency>
```


### 3 required topics

- ✓




[Introduction to Spring Web MVC](#)

▼
- ✓



[Web security, OWASP](#)

▼
- ✓



[Authentication and Authorization](#)

▼

### 2 dependent topics

- [Web](#)

▼
- 

[Authentication](#)

▼

This dependency includes autoconfiguration and by default enables security-related features like HTTP basic authentication, form-based authentication, protection against CSRF, and other things. It will be easier to get acquainted with Spring Security by looking at some examples first. So let's create a simple app, run it, and see what happens!

## §2. Preparation

Let's assume that we started a new Spring Boot project, added a web and security starter dependency, and created a simple `index.html` file located in the `/resources/static` folder.

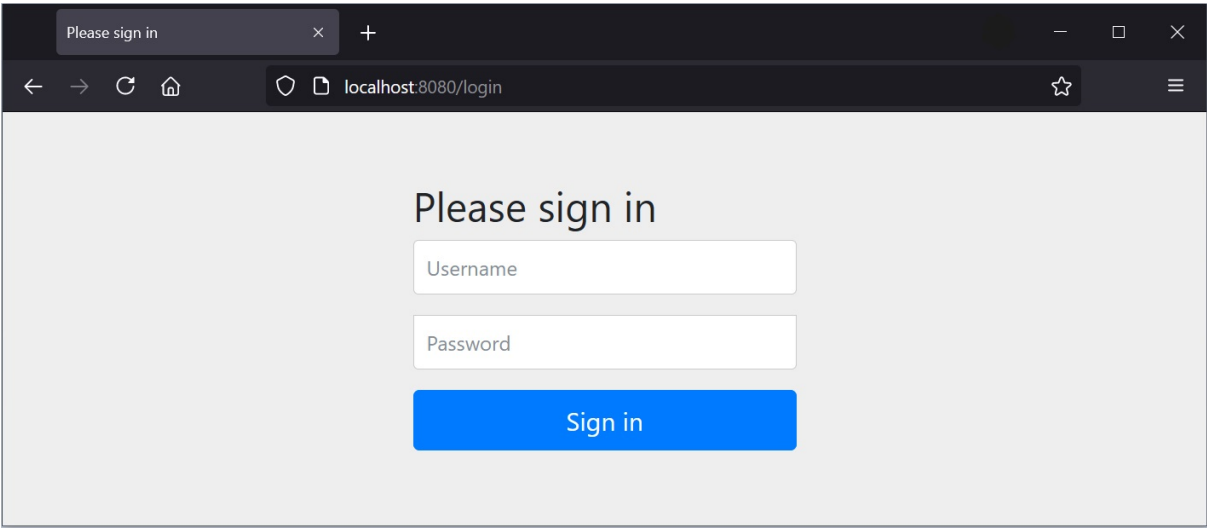
Here's the content of the file:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Security Test</title>
6 </head>
7 <body>
8   <h1>Access granted!</h1>
9 </body>
10 </html>
```

Now let's run the app and try to access the file.

## §3. Secured app

If we try to access the file by pointing our web browser to `http://localhost:8080/`, instead of seeing the expected "Access granted!" message, we'll be redirected to `http://localhost:8080/login` and shown the following form:



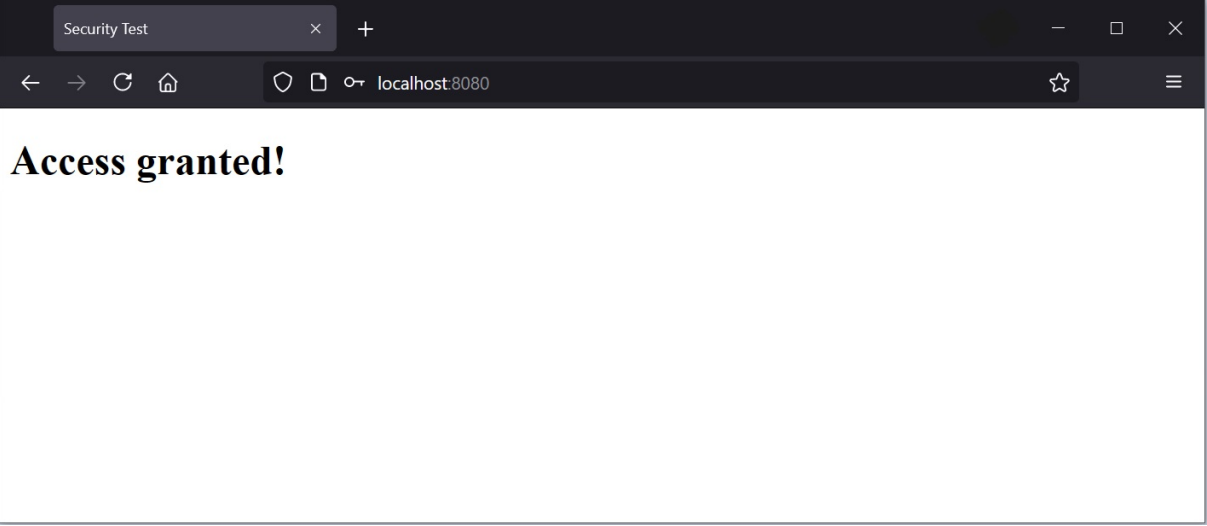
Wait a minute, we haven't implemented any login form, haven't created any users... And what are our login and password?

Our app is secured once we have added the dependency. This form is provided by autoconfiguration, and Spring Security also generated a default user. To access our app, we need to enter the correct login and password. The default login is "user", and we can find the password in the console log. The log entry will look something like this:

```
1 Using generated security password: acfa1db0-9ecf-4edf-b0a6-33d5199a8091
```

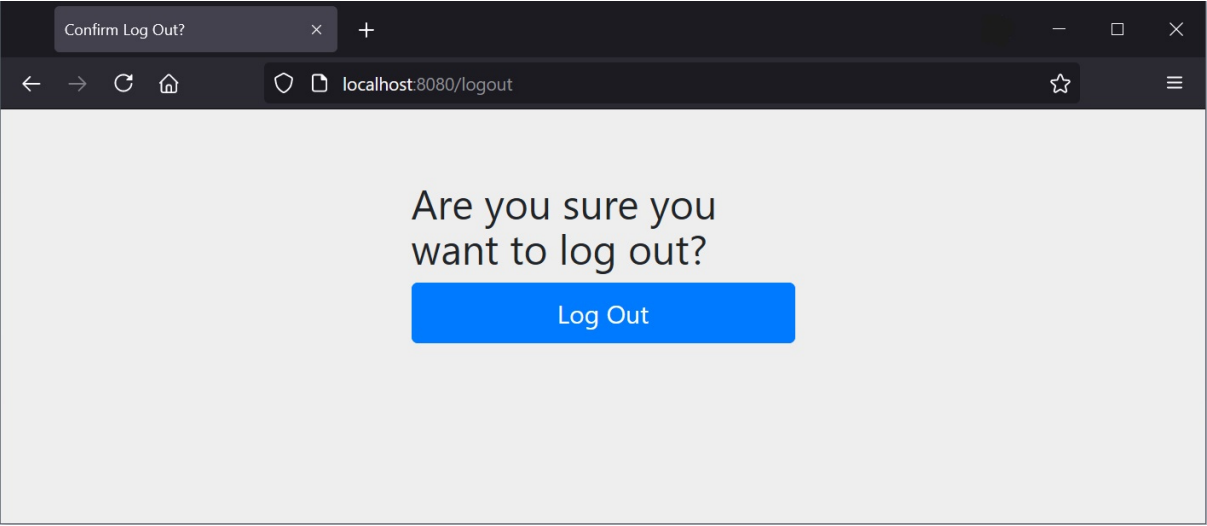
The default login is always "user" (case-insensitive) but every time we run an app a new password is generated. So your password will be different every time.

If we enter the generated login and password we'll be redirected from `http://localhost:8080/login` back to the URL that we were trying to access, `http://localhost:8080/`, and we'll see our page:



Autoconfiguration also adds a default logout page.

If we input `http://localhost:8080/logout` URL in a web browser we'll see the following page:



Isn't it quite a lot for just one dependency without any code added? You'll learn more about this in the upcoming topics. For now, let's see how we can set our own login and password.

## §4. Configuration

We can change the login and password in the `application.properties` file.

Here's how you can specify the login:

```
1 spring.security.user.name=someone
```

And you can set the password like this:

```
1 spring.security.user.password=123
```

If we change the password and reload the server, the default password won't be generated. Also, the console will not show the password anymore.

## §5. Conclusion

In this topic, you've got acquainted with Spring Security and we've created a simple secured app. We've seen that starter dependency enables mandatory authentication for URLs, adds the login form and logout, and creates a default user. You've also learned how to change your login and password using the `application.properties` file.

Spring Boot Security autoconfiguration is a great way to get started with security, but most applications will demand explicitly configuring security to meet their

Table of contents:

unique security requirements. In the upcoming topics, you'll learn how to add multi-user support to an application, secure specific URLs, create a custom login and logout, and other interesting features of this framework.

Now, let's solve some tasks!

119 users liked this piece of theory. 1 didn't like it. What about you?

😊

🙂

😐

😞

😡

Start practicing

Verify to skip

Report a typo

1 Getting started with Spring Security.

§1. Dependency

§2. Preparation

§3. Secured app

§4. Configuration


§5. Conclusion

Discussion

Comments (7)

Useful links (1)

Show discussion

JetBrains Academy

Tracks

Pricing

For organizations

About

Contribute



Careers


Become beta tester

Be the first to see what's new

Terms

Support



Made with  by Hyperskill and JetBrains

https://hyperskill.org/learn/step/13856

4/4