Spring Security

Truong Nhat Hoang – 51703092
Huynh Van Duy – 51703006
Tran Quoc Linh - 51703124

What is Spring Security?

Spring Security is a Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications.

Feature

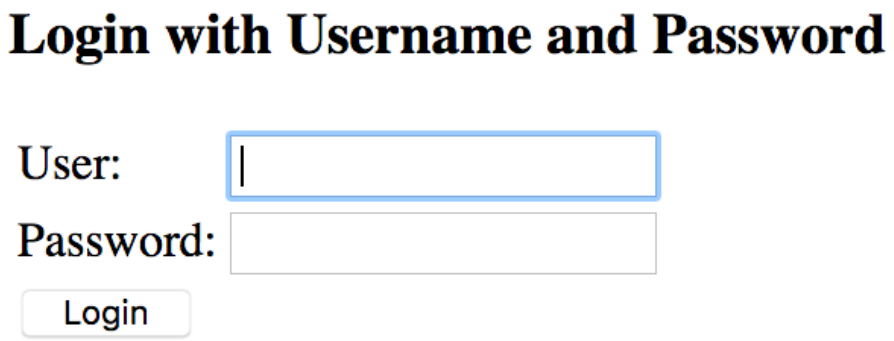
- Comprehensive and extensible support for both Authentication and Authorization
- Protection against attacks like session fixation, clickjacking, cross site request forgery, etc.
- Servlet API integration
- Optional integration with Spring Web MVC
- …

When do we use?

Base on spring security feature, we can use it when we want to hide certain links or buttons based on user's role so that they will not be able to access that functionality.

Where do we use?

Spring security can be used for authentication and authorization purposes in your application (such as login). You can secure you app with it. Authenticate user for web apps, mobile apps, etc. It provides integration with LDAP as well.



Why spring-security?

Spring Security fill a gap in the universe of Java third-party libraries. Standards such as Java Authentication and Authorization Service (JAAS) or Java EE Security do offer some ways of performing some of the same authentication and authorization functions, but Spring Security includes everything you need to implement a top-to-bottom application security solution in a concise and sensible way.

Why spring-security?

Additionally, Spring Security offers out-of-the-box integration with many common enterprise authentication systems; so it's adaptable to most situations with little effort (beyond configuration) on the part of the developer.

There's really no other mainstream framework good like it!

Build a simple security web application

Requirements and start the project

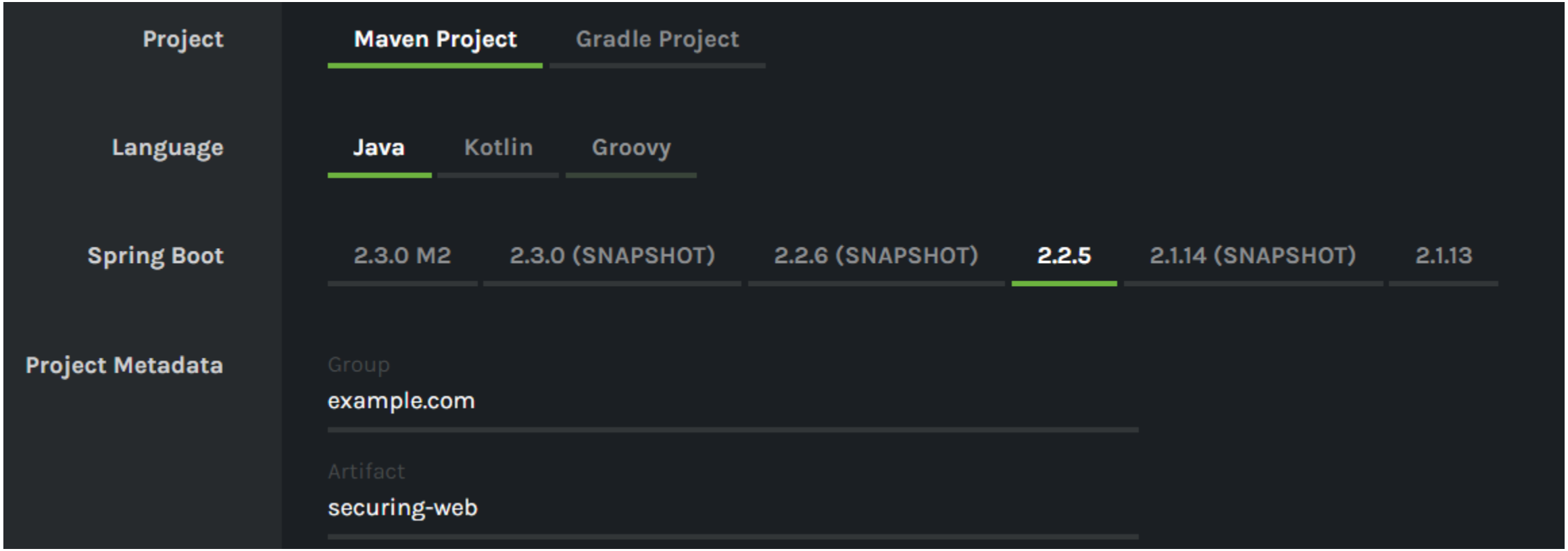
Requirement

- JDK 1.8 or later
- Gradle 4+ or Maven 3.2+

Then, starting with Spring Initializr

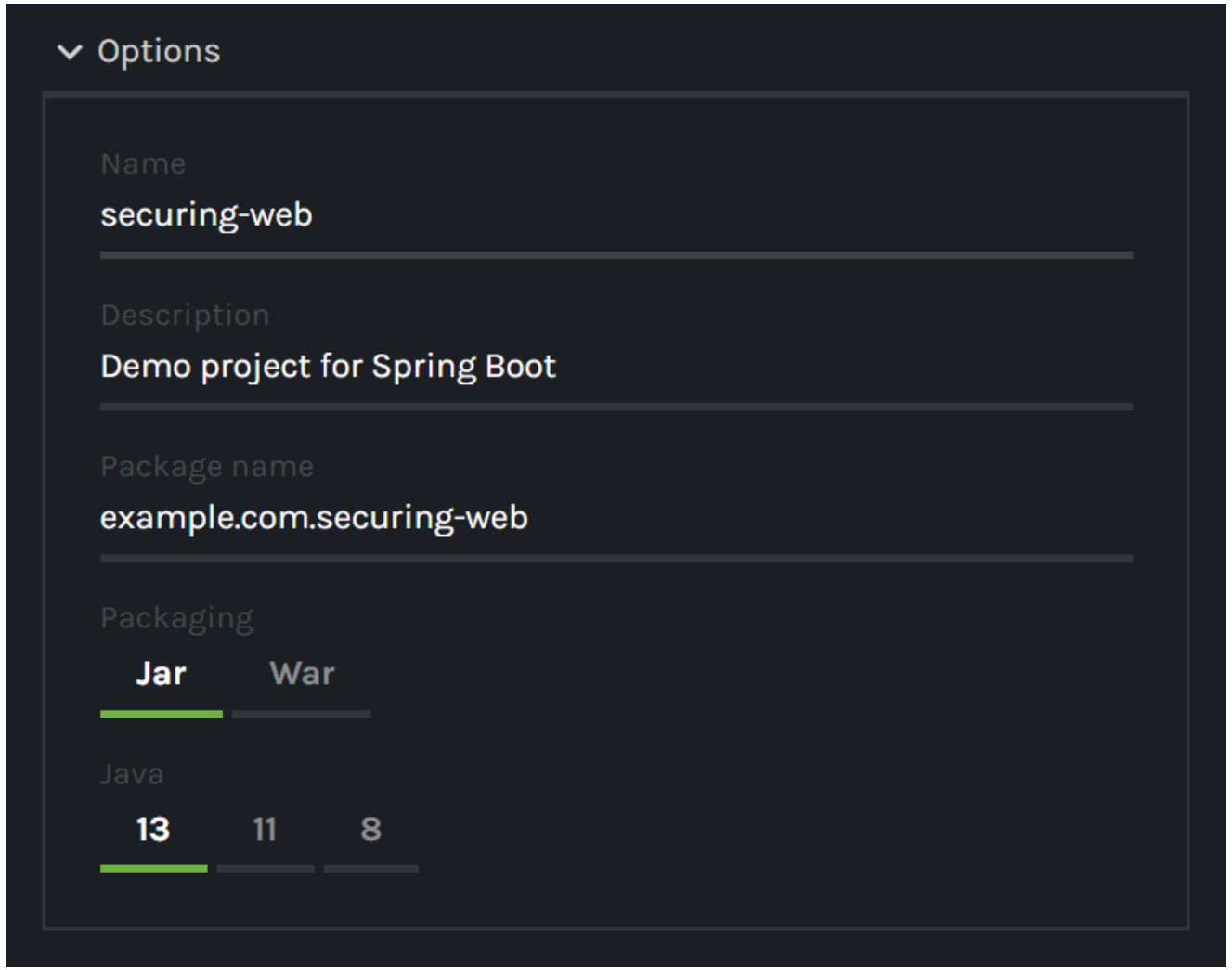
Start with Spring Initializr

Select compatible options with your environment



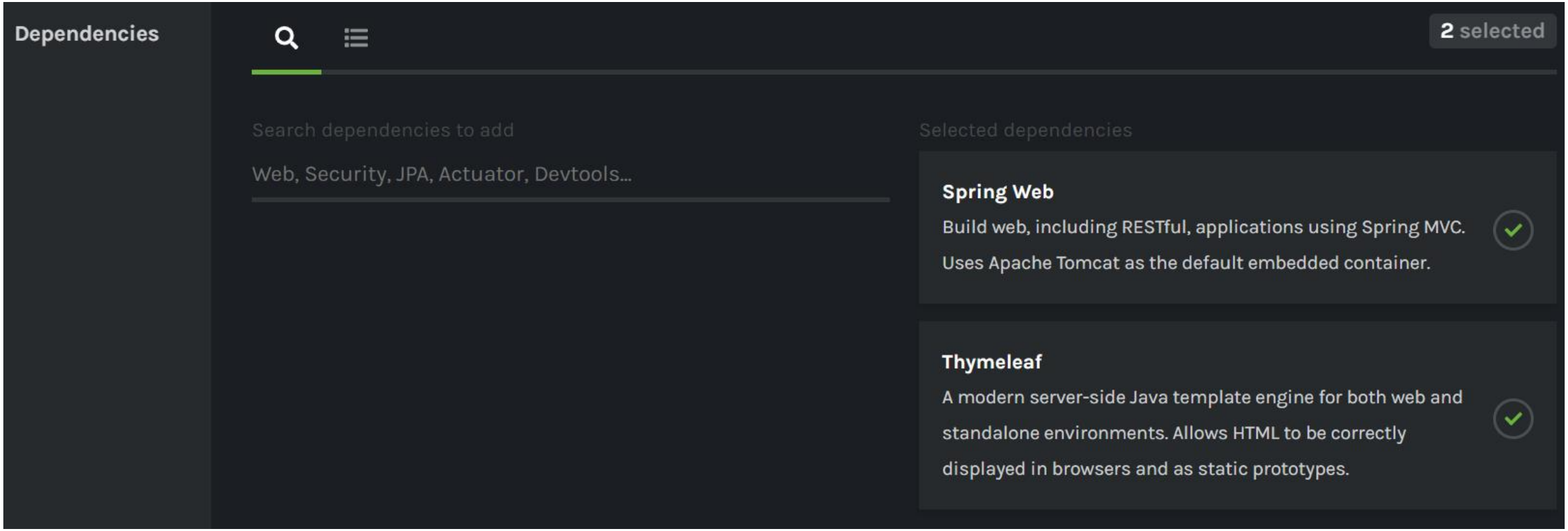
Start with Spring Initializr

Select compatible options with your environment



Start with Spring Initializr

Add two dependence



Then click on generate button

Create an Unsecured Web Application

Create two HTML files (home.html and hello.html) follow path

- src/main/resources/templates/home.html
- src/main/resources/templates/hello.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-
extras-springsecurity3">
    <head>
        <title>Spring Security Example</title>
    </head>
    <body>
        <h1>Welcome!</h1>
        <p>Click <a th:href="@{/hello}">here</a> to see a
              greeting.</p>
    </body>
</html>
```

home.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-
springsecurity3">
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1>Hello world!</h1>
    </body>
</html>
```

hello.html

Create an Unsecured Web Application

And a MvcConfig.java file from src/main/java/com/example/securingweb/MvcConfig.java

```
package com.example.securingweb;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration public class MvcConfig implements WebMvcConfigurer {
    public void addViewControllers(ViewControllerRegistry registry) {
            registry.addViewController("/home").setViewName("home");
            registry.addViewController("/").setViewName("home");
            registry.addViewController("/hello").setViewName("hello");
            registry.addViewController("/login").setViewName("login");
    }
}
```

Set up Spring Security

Add two extra entries (one for the application and one for testing) to the <dependencies> element in pom.xml

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

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

Set up Spring Security

Create login.html file and modify hello.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
    <head> <title>Spring Security Example </title> </head>
    <body>
        <div th:if="${param.error}"> Invalid username and password. </div>
        <div th:if="${param.logout}"> You have been logged out. </div>
        <form th:action="@{/login}" method="post">
        <div><label> User Name : <input type="text" name="username"/>
</label></div>
        <div><label> Password: <input type="password" name="password"/>
</label></div>
        <div><input type="submit" value="Sign In"/></div>
        </form>
     </body>
</html>
```

login.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-
springsecurity3">
    <head> <title>Hello World!</title> </head>
    <body>
        <h1 th:inline="text">Hello
[[${#httpServletRequest.remoteUser}]]!</h1>
        <form th:action="@{/logout}" method="post">
            <input type="submit" value="Sign Out"/>
        </form>
    </body>
</html>
```

hello.html

Set up Spring Security

Then add two java file below to path src/main/java/com/example/securingweb/

- WebSecurityConfig.java (file attach)
- SecuringWebApplication.java (file attach)

You need not to modify them.

Run the Application

Open cmd and cd to project folder. Using one of following to build an executable JAR

- `mvnw spring-boot:run`
- `mvnw clean package`

Then, run jar just build with

```
java –jar target/<filename>.jar
```

Finally, open browser with url http://localhost:8080/ to see the result.

Advantages and disadvantages

Advantages:

- It is very easy to develop Spring Based applications with Java or Groovy.
- It reduces a lot of development time and increases productivity.
- It avoids writing of boilerplate Code, Annotations and XML Configuration.
- It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.

Advantages and disadvantages

Advantages:

- It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
- It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.
- It provides CLI (Command Line Interface) tool to develop and test Spring Boot (Java or Groovy) Applications from command prompt very easily and quickly.
- It provides lots of plugins to develop and test Spring Boot Applications and it uses Build Tools like Maven and Gradle
- It provides lots of plugins to work with embedded and in-memory Databases very easily.

Advantages and disadvantages

Disadvantages:

It is very tough and time consuming process to convert existing or legacy Spring Framework projects into Spring Boot Applications. It is applicable only for brand new/Greenfield Spring Projects.

References i

[1] https://spring.io/projects/spring-security

[2] https://en.wikipedia.org/wiki/Spring_Security

[3] https://www.youtube.com/watch?v=sm-8qfMWEV8

[4] Mick Knutson, Robert Winch, Peter Mularien - Spring Security-Packt Publishing (2017)

References ii

[5] https://kipalog.com/posts/Huong-dan-lap-trinh-Spring-Security

[6] https://www.developer.com/java/ent/what-is-spring-security.html

[7] https://spring.io/guides/gs/securing-web/

Thanks