**Защита веб-приложения**

Это руководство проведет вас через процесс создания простого веб-приложения с ресурсами, защищенными Spring Security.

Что вы будете строить

Вы создадите приложение Spring MVC, которое защищает страницу с помощью формы входа, поддерживаемой фиксированным списком пользователей.

Как заполнить это руководство

Как и в большинстве [руководств по началу работы с](https://spring.io/guides) Spring , вы можете начать с нуля и выполнить каждый шаг или обойти основные шаги настройки, которые вам уже знакомы. В любом случае вы получите рабочий код.

Чтобы **начать с нуля** , перейдите к [разделу «Запуск с Spring Initializr»](https://spring.io/guides/gs/securing-web/#scratch) .

Чтобы **пропустить основы** , сделайте следующее:

* [Загрузите](https://github.com/spring-guides/gs-securing-web/archive/main.zip) и распакуйте исходный репозиторий для этого руководства или клонируйте его с помощью [Git](https://spring.io/understanding/Git) :git clone <https://github.com/spring-guides/gs-securing-web.git>
* cd в gs-securing-web/initial
* Забегайте вперед, чтобы [создать незащищенное веб-приложение](https://spring.io/guides/gs/securing-web/#initial) .

**Когда вы закончите** , вы можете сравнить свои результаты с кодом в gs-securing-web/complete.

Начиная с Spring Initializr

Если вы используете Maven, посетите [Spring Initializr,](https://start.spring.io/" \l "!type=maven-project&language=java&platformVersion=2.4.3.RELEASE&packaging=jar&jvmVersion=1.8&groupId=com.example&artifactId=securing-web&name=securing-web&description=Demo%20project%20for%20Spring%20Boot&packageName=com.example.securing-web&dependencies=web,thymeleaf) чтобы сгенерировать новый проект с необходимыми зависимостями (Spring Web и Thymeleaf).

В следующем листинге показан pom.xmlфайл, который создается при выборе Maven:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>securing-web-initial</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>securing-web-initial</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Создать незащищенное веб-приложение

Прежде чем вы сможете применить безопасность к веб-приложению, вам необходимо защитить веб-приложение. В этом разделе вы узнаете, как создать простое веб-приложение. Затем вы защитите его с помощью Spring Security в следующем разделе.

Веб-приложение включает два простых представления: домашнюю страницу и страницу «Hello, World». Домашняя страница определена в следующем шаблоне Thymeleaf (из src/main/resources/templates/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>Spring Security Example</title>

</head>

<body>

<h1>Welcome!</h1>

<p>Click <a th:href="@{/hello}">here</a> to see a greeting.</p>

</body>

</html>

Это простое представление включает ссылку на /helloстраницу, которая определена в следующем шаблоне Thymeleaf (из 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>Hello World!</title>

</head>

<body>

<h1>Hello world!</h1>

</body>

</html>

Веб-приложение основано на Spring MVC. В результате вам необходимо настроить Spring MVC и настроить контроллеры представления для предоставления этих шаблонов. В следующем листинге (из src/main/java/com/example/securingweb/MvcConfig.java) показан класс, который быстро настраивает контроллеры Spring MVC в приложении:

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");

}

}

addViewControllers()Метод (который переопределяет метод того же имени в WebMvcConfigurer) добавляет четыре контроллера представления. Два контроллера представления ссылаются на представление, имя которого home(определено в home.html), а другой ссылается на представление с именем hello(определено в hello.html). Четвертый контроллер представления ссылается на другое представление с именем login. Вы создадите это представление в следующем разделе.

На этом этапе вы можете перейти к « [Запустить приложение](https://spring.io/guides/gs/securing-web/#run_the_app) » и запустить приложение, не входя ни в какую систему.

Теперь, когда у вас есть незащищенное веб-приложение, вы можете добавить к нему безопасность.

Настроить Spring Security

Предположим, вы хотите запретить неавторизованным пользователям просматривать страницу приветствия по адресу /hello. Как и сейчас, если посетители переходят по ссылке на домашней странице, они видят приветствие без каких-либо препятствий, которые могли бы их остановить. Вам нужно добавить барьер, который заставит посетителя войти в систему, прежде чем он сможет увидеть эту страницу.

Вы делаете это, настраивая Spring Security в приложении. Если Spring Security находится в пути к классам, Spring Boot [автоматически защищает все конечные точки HTTP](https://docs.spring.io/spring-boot/docs/2.4.3/reference/htmlsingle/#boot-features-security) с помощью «базовой» аутентификации. Однако вы можете дополнительно настроить параметры безопасности. Первое, что вам нужно сделать, это добавить Spring Security в путь к классам.

В Maven вам нужно добавить две дополнительные записи (одну для приложения и одну для тестирования) в <dependencies>элемент pom.xml, как показано в следующем листинге:

<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>

В следующем листинге показан готовый pom.xmlфайл:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>securing-web-complete</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>securing-web-complete</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<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>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Следующая конфигурация безопасности (от src/main/java/com/example/securingweb/WebSecurityConfig.java) гарантирует, что только аутентифицированные пользователи могут видеть секретное приветствие:

package com.example.securingweb;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.antMatchers("/", "/home").permitAll()

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login").permitAll()

.and()

.logout()

.permitAll();

}

//такой механизм подразумевает сохранение логина/пароля в памяти компьютера (.inMemoryAuthentication())

@Bean

@Override

public UserDetailsService userDetailsService() {

UserDetails user =

User.withDefaultPasswordEncoder()

.username("user")

.password("password")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

WebSecurityConfigКласс помечается @EnableWebSecurityвключить поддержку веб - безопасности Spring Security и обеспечивают интеграцию Spring MVC. Он также расширяет WebSecurityConfigurerAdapterи переопределяет несколько своих методов, чтобы установить некоторые особенности конфигурации веб-безопасности.

В configure(HttpSecurity)методе определяет , какие пути URL должны быть надежно закреплены и которые не должны. В частности, /и /homeпуть настроены , чтобы не требовать никакой аутентификации. Все остальные пути должны быть аутентифицированы.

Когда пользователь успешно входит в систему, он перенаправляется на ранее запрошенную страницу, требующую аутентификации. Существует настраиваемая /loginстраница (указанная с помощью loginPage()), и все могут ее просматривать.

Метод userDetailsService() настраивает в памяти компьютера аутентификацию (.inMemoryAuthentication()) с одним пользователем. Этому пользователю дается имя пользователя user, пароль passwordи роль USER.

Теперь вам нужно создать страницу входа. Уже существует контроллер представления для login, поэтому вам нужно только создать само представление входа в систему  src/main/resources/templates/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>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>

Этот шаблон Thymeleaf представляет собой форму, которая вводит имя пользователя и пароль и отправляет их /login. В соответствии с настройками Spring Security предоставляет фильтр, который перехватывает этот запрос и аутентифицирует пользователя. Если пользователю не удается пройти аутентификацию, страница перенаправляется на /login?error, и на вашей странице отображается соответствующее сообщение об ошибке. После успешного выхода ваше приложение будет отправлено по адресу /login?logout, и на вашей странице отобразится соответствующее сообщение об успешном завершении.

Наконец, вам необходимо предоставить посетителю возможность отобразить текущее имя пользователя и выйти из системы. Для этого в src/main/resources/templates/hello.html обновите файл hello.html чтобы поздороваться с текущим пользователем и содержать Sign Outформу:

<!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>

Мы отображаем имя пользователя, используя интеграцию Spring Security с HttpServletRequest#getRemoteUser(). Форма «Выйти» отправляет сообщение POST на адрес /logout. После успешного выхода из системы он перенаправляет пользователя на /login?logout.

Запустить приложение

Spring Initializr создает для вас класс приложения. В этом случае вам не нужно изменять класс. В следующем листинге (из src/main/java/com/example/securingweb/SecuringWebApplication.java) показан класс приложения:

package com.example.securingweb;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SecuringWebApplication {

public static void main(String[] args) throws Throwable {

SpringApplication.run(SecuringWebApplication.class, args);

}

}

Создайте исполняемый JAR

Вы можете запустить приложение из командной строки с помощью Gradle или Maven. Вы также можете создать один исполняемый файл JAR, содержащий все необходимые зависимости, классы и ресурсы, и запустить его. Создание исполняемого файла jar упрощает доставку, версию и развертывание службы как приложения на протяжении всего жизненного цикла разработки, в различных средах и т. Д.

Если вы используете Gradle, вы можете запустить приложение, используя ./gradlew bootRun. В качестве альтернативы вы можете создать файл JAR, используя, ./gradlew buildа затем запустить файл JAR, как показано ниже:

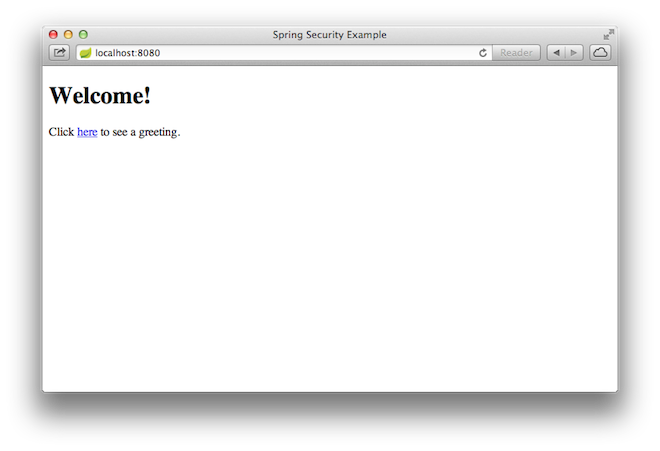
java -jar build / libs / gs-securing-web-0.1.0.jar

Если вы используете Maven, вы можете запускать приложение, используя ./mvnw spring-boot:run. В качестве альтернативы вы можете создать файл JAR, ./mvnw clean packageа затем запустить файл JAR следующим образом:

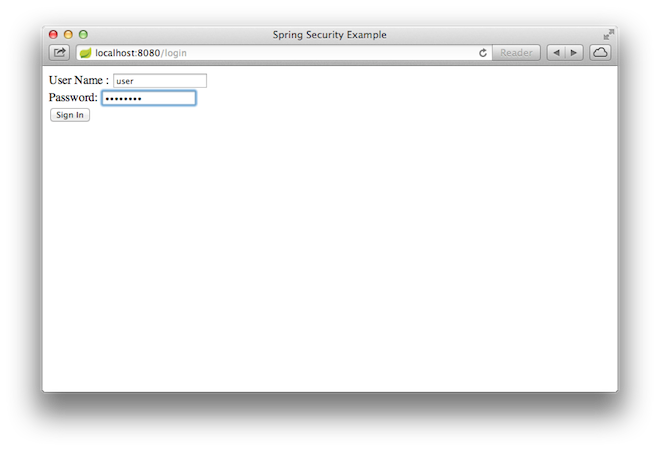
java -jar цель / gs-securing-web-0.1.0.jar

|  |  |
| --- | --- |
|  | Описанные здесь шаги создают работающий JAR. Вы также можете [создать классический WAR-файл](https://spring.io/guides/gs/convert-jar-to-war/) . |

После запуска приложения укажите в браузере [http://localhost:8080](http://localhost:8080/). Вы должны увидеть домашнюю страницу, как показано на следующем изображении:

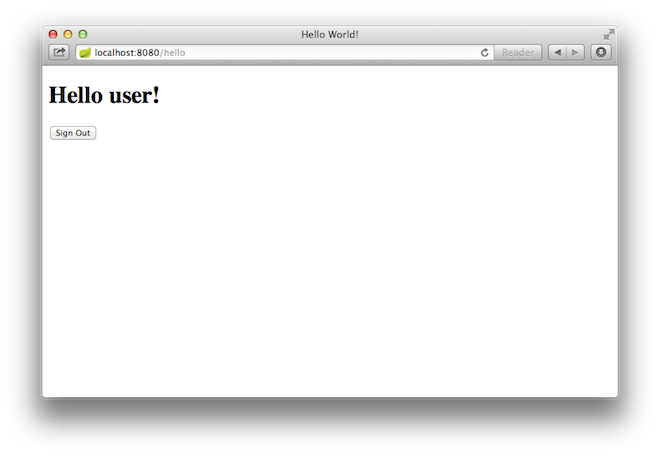


Когда вы нажимаете на ссылку, она пытается перенаправить вас на страницу приветствия по адресу /hello. Однако, поскольку эта страница защищена и вы еще не вошли в систему, она переводит вас на страницу входа, как показано на следующем изображении:



|  |  |
| --- | --- |
|  | Если вы перепрыгнули сюда с незащищенной версией, вы не увидите страницу входа. Вы должны сделать резервную копию и написать остальной код, основанный на безопасности. |

На странице входа войдите как тестовый пользователь, введя userи passwordв поля имени пользователя и пароля, соответственно. После отправки формы входа вы аутентифицируетесь, а затем переходите на страницу приветствия, как показано на следующем изображении:



Если вы нажмете кнопку « **Выйти»** , ваша аутентификация будет отменена, и вы вернетесь на страницу входа с сообщением о том, что вы вышли из системы.

Резюме

Поздравляю! Вы разработали простое веб-приложение, защищенное Spring Security.