**Spring Security за 15 минут**

**Что вы сделаете?**

Вы интегрируете web-app на Spring Boot с библиотекой Spring Security. Мы возьмем незащищенное приложение, которое мы разбирали в прошлой статье и защитим его с помощью Spring Security:

* <https://smartcoderteam.github.io/spring-boot-simple-app-in-15-minutes/>
* <https://github.com/levrun/spring-boot-example>

**Что вам понадобится?**

* Около 15-50 минут
* Ваш любимый текстовый редактор или IDE
* JDK 1.8 или выше
* Maven 3.0+

**Как пользоваться этой статьей**

* Внимательно прочитайте статью
* Скачайте исходный код с github
* Запустите и проверьте работу
* Удостоверьтесь что код вам понятен
* Если что то осталось неясно, смело задавайте вопросы мне или *google*

**Первичная настройка Spring-Security**

Давайте представим, что нам хочется предотвратить неавторизованный доступ к страничке <http://localhost:8080/hello>. Сейчас если пользователь нажмет ссылку на базовой странице <http://localhost:8080/base> ничто не остановит его. Вы должны добавить какой то *барьер* который бы заставлял пользователя авторизоваться перед тем как увидеть эту страничку. Вы сделаете это сконфигурировав Spring Security. Если Spring Security находится в classpath [Spring Boot автоматически защищает все HTTP-endpoints базовой(basic) авторизацией](http://docs.spring.io/spring-boot/docs/1.4.2.RELEASE/reference/htmlsingle/" \l "boot-features-security)

В дальнейшем вы сможете более тонко настроить security. Первое что вам надо сделать это добавить Spring Security в classpath:

<dependency>

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

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

</dependency>

**WebSecurityConfigurerAdapter**

@Configuration

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

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

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.permitAll();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth

.inMemoryAuthentication()

.withUser("user").password("password").roles("USER");

}

}

* *@EnableWebSecurity* включает поддержку web security и обеспечивает интеграцию со Spring MVC
* *сonfigure(HttpSecurity)* определяет какой URL нужно защищать, а какой не надо(по умолчанию все защищено)
* *.antMatchers(“/”, “/base”).permitAll()* данные urls защищать не надо(*.permitAll()*)
* *.loginPage(“/login”)* определяется страница для совершения логина в приложение, которая доступна всем
* *.logout().permitAll()* определяет что разлогиниться могут все
* *configureGlobal(AuthenticationManagerBuilder)* устанавливает in-memory хранилище пользователей с одним пользователем

**Создадим login-страницу**

Добавим файл *src/main/resources/templates/login.html* с шаблоном *thymeleaf* в котором включим интеграцию со Spring-Security - [*thymeleaf-extras-springsecurity3*](https://github.com/thymeleaf/thymeleaf-extras-springsecurity)

<html xmlns="<http://www.w3.org/1999/xhtml>" xmlns:th="[http://www.thymeleaf.org](http://www.thymeleaf.org/)"

xmlns:sec="<http://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>

И не забудем добавить в *MvcConfig* для страницы логина натройку контроллера - *registry.addViewController(“/login”).setViewName(“login”);*

А также изменим страничку *hello* добавив туда ссылку для разлогинивания:

<html xmlns="<http://www.w3.org/1999/xhtml>" xmlns:th="[http://www.thymeleaf.org](http://www.thymeleaf.org/)"

xmlns:sec="<http://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 показываем имя пользователя на страничке благодаря использованию интеграции со Spring Security *HttpServletRequest#getRemoteUser()* Ссылка “Sign Out” создаст POST запрос к “/logout” и далее переведет пользователя на “/login?logout”. Благодаря чему пользователь увидит это сообщение:

<div th:if="${param.logout}">

You have been logged out.

</div>

**Как это устроено?**

Эта простейшая авторизация работает благодаря [HTTP\_cookie](https://en.wikipedia.org/wiki/HTTP_cookie). Рекомендую отследить момент до логина и после и посмотреть на состояние cookie с именем JSESSIONID. Вы увидете что после логина cookie не меняет значение. После разлогинивания это значение поменяется, что и будет означать что пользователь вышел из системы.

В chrome нажмите Ctrl-Shift-I -> Network tab и далее заголовки httml запроса

Тоесть как только пользователь залогинился Spring Security присваивает залогиненому пользователю отдельный JSESSIONID по которому он определяет его при последующих запросах. Это все похоже на работу сессий в сервлетах. Подробнее прочитайте вот эту ссылку [how-does-spring-security-sessions-work](http://stackoverflow.com/questions/8425856/how-does-spring-security-sessions-work)

**Запуск**

* Запустите main-метод
* Откройте в браузере [http://localhost:8080](http://localhost:8080/)
* Попытайтесь зайти на <http://localhost:8080/hello> не вводя имя и пароль. Убедитесь что вы не сможете это сделать.
* Попробуйте зайти введя правильный логин и пароль на страничке <http://localhost:8080/login>