

Informatyka, Aplikacje internetowe i mobilne, semestr 5

Programowanie aplikacji Webowych

Laboratorium nr 7 - uzupełnienie

Tworzenie aplikacji Webowej SpringBooks z wykorzystaniem frameworka Spring Boot

Zabezpiecz aplikację SpringBooks dodając listę użytkowników, możliwość ich rejestracji i logowania oraz ogranicz możliwość edycji i kasowania danych przydzielając prawa do tych operacji tylko użytkownikom zalogowanym.

1. Uzupełnienie projektu Gradle

Do projektu Gradle aplikacji SpringBook dodaj zależność Spring Security. Zależności można dodawać wybierając z podręcznego menu projektu Spring | Add Starters. W oknie New Spring Starter Project Dependencies wybierz wszystkie oraz dodatkowo Spring Security. W kolejnym oknie wprowadź odpowiednie zmiany dodając 3 zaproponowane zależności.

W pliku **build.gradle**, w sekcji dependencies powinny się znaleźć następujące linie:

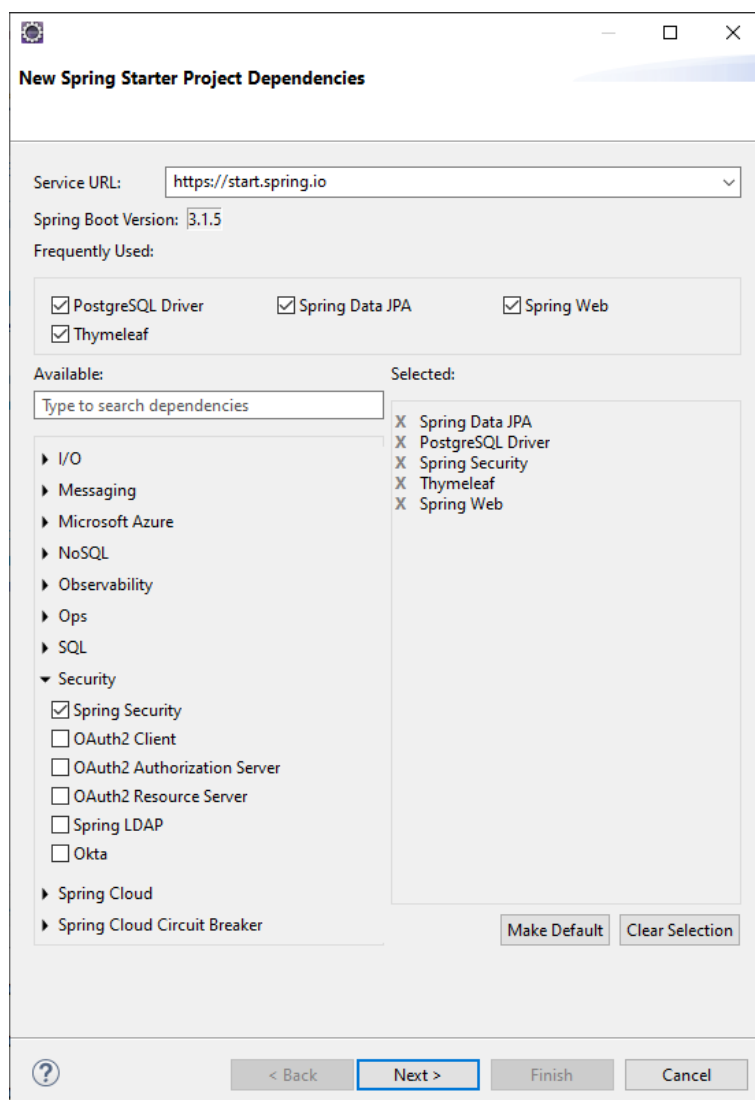
```
implementation
'org.springframework.boot:spring-
boot-starter-security'

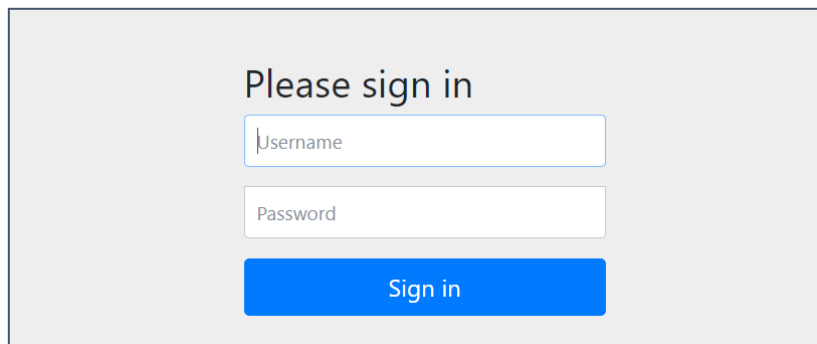
implementation
'org.thymeleaf.extras:thymeleaf-
extras-springsecurity6'

testImplementation
'org.springframework.security:sprin
g-security-test'
```

Sprawdź działanie aplikacji i dostęp do różnych stron.

Pakiet startowy bezpieczeństwa spring-boot-starter-security uzupełnił automatycznie kod aplikacji i zablokował dostęp do wszystkich stron wyświetlając podstawową stronę do logowania. Może to być podstawowe zabezpieczenie dla dowolnej aplikacji.





Domyślny login jest teraz ustawiony jako user, hasło jest natomiast generowane losowo i powinno się pojawić w logu aplikacji wyświetlanym w konsoli w postaci:

Using generated security password: dbdb0988-a47b-43bd-b066-3225b0832954

Jeśli w konsoli nie ma podobnej informacji dodaj komentarz w pliku **application.properties** w linii z ustawieniem `logging.level.org.springframework=ERROR`, ponownie uruchom aplikację i sprawdź możliwość zalogowania.

2. Podstawowa konfiguracja bezpieczeństwa

W katalogu `springbooks` dodaj nowy folder do implementacji bezpieczeństwa o nazwie `security`.

W pakiecie `security` wygeneruj klasę do konfiguracji bezpieczeństwa o nazwie `SecurityConfig`. Żeby klasa pełniła rolę konfiguracyjnej trzeba ustawić jej adnotację `@Configuration`. Żeby zapewnić bezpieczeństwo dla aplikacji Web, a jednocześnie integrację z MVC trzeba dodać do tej klasy jeszcze jedną adnotację `@EnableWebSecurity`.

Wewnątrz klasy potrzebna będzie metoda z adnotacją `@Bean` do wstrzyknięcia ziarenka Javy `SecurityFilterChain`. Definiuje ona które URLe aplikacji zostaną zabezpieczone, a które nie. Zwykle ścieżka do katalogu głównego i `/home` lub `/index` pozostaje niezabezpieczona. Dodaj do klasy `SecurityConfig` kod, który nie pozwoli edytować ani dodawać nowych książek niezalogowanym użytkownikom:

```
@Bean
SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(authorize -> authorize
            .requestMatchers("/new_book", "/edit_book/**")
            .authenticated()
            .anyRequest().permitAll()
        )
        .formLogin(Customizer.withDefaults());
    return http.build();
}
```

Gdy użytkownik poprawnie się zaloguje, zostanie automatycznie przekierowany na URL, którego żądał wcześniej. Metoda `formLogin` wskazuje `withDefaults` czyli zachowanie zdefiniowane domyślnie w `SpringSecurity`.

W przypadku aplikacji SpringBooks wszyscy powinni mieć dostęp do strony głównej i list czyli widoków w plikach ..._list, natomiast pozostałe strony powinny wymagać zalogowania. Sprawdź czy podane ścieżki odpowiadają żądaniom w Twoim SpringBooks oraz działanie aplikacji. Uzupełnij listę adresów wymagających autoryzacji i ponownie sprawdź działanie aplikacji.

3. Definiowanie użytkowników

Użytkowników aplikacji można zapisać w kodzie i może to być dobrym rozwiązaniem gdy jest ich niewielu i rzadko się zmieniają, albo na etapie tworzenia aplikacji. W większości przypadków użytkowników trzeba zapisywać w pliku lub w bazie danych.

Na początek wygenerujemy jednego użytkownika do testowania i zapiszemy go w pamięci. Służy do tego klasa UserDetails oraz UserDetailsService implementowana w klasie InMemoryUserDetailsManager Kod dla użytkownika o loginie user1 i hasle passu1 (bez kodowania {noop}) może wyglądać następująco:

```
@Bean
UserDetailsService userDetailsService(){
    UserDetails user = User.builder()
        .username("user1")
        .password("{noop}passu1")
        .roles("USER")
        .build();
    return new InMemoryUserDetailsManager(user);
}
```

Podczas uwierzytelniania i autoryzacji korzystamy z wielu gotowych klas Spring Security. Trzeba je jednak uzupełniać kodami charakterystycznymi dla naszej aplikacji. Jedną z takich klas jest właśnie UserDetails widoczna powyżej. Analogicznie, generując nowe obiekty UserDetails, można tutaj dodać innych stałych użytkowników.

Zwykle hasła są kodowane. W klasie potrzebna będzie do tego metoda generująca statyczny obiekt klasy PasswordEncoder z ustawieniem wybranego kodowania. Klasa z ustawieniem kodowania BCrypt może mieć postać:

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

Spring Security udostępnia również inne rodzaje kodowania, w tym zgodne wstecz, w tym również takie, które nie są już uważane za bezpieczne. Dostępne kodowania to BCrypt, Argon2, Pbkdf2 i CScrypt oraz niezalecane SScrypt i Standard.

Zmień hasło dla user1 na "\$2y\$10\$69spfiVfvE2TTx83aibCmuhzVTJMV5NM3jqJjDAeN9F6aM0cVmdD2" i sprawdź działanie aplikacji.

4. Własny formularz logowania

Standardowy formularz do logowania można zamienić na własny, dedykowany dla aplikacji. Potrzebny jest do tego szablon z kodem HTML i Thymeleaf. Dodaj do aplikacji szablon o nazwie login.html i umieść

w nim kod formularza. Domyślny mechanizm Spring Security wymaga żeby w formularzu znalazły się standardowe pola o nazwach username i password. Przykładowy kod podano poniżej.

```
<form action="#" th:action="@{/login}" method="post">
  <table>
    <tr>
      <td>Login</td>
      <td><input type="text" id="username" name="username"/></td>
    </tr>
    <tr>
      <td>Password</td>
      <td><input type="password" id="password" name="password"/></td>
    </tr>
  </table>
</form>
```

Zdefiniuj klasę LoginController z metodą do wyświetlania formularza o nazwie showFormLogin z mapowaniem @GetMapping("/login") oddającą w wyniku "login".

Pozostaje jeszcze przekserowanie obsługi logowania z domyślnego formularza do naszego. W tym celu w pliku SecurityConfig, w metodzie securityFilterChain trzeba zmienić parametr metody formLogin na następującą postać:

```
.formLogin(form -> form
    .loginPage("/login")
    .loginProcessingUrl("/login")
    .defaultSuccessUrl("/")
    .permitAll());
```

5. Nawigacja – opcje w menu

W menu aplikacji Spring Books powinny się znaleźć pola Login i Logout wyświetlane na zmianę w zależności od tego czy użytkownik został uwierzytelniony czy nie. Służy do tego parametr sec:authorize, który daje dostęp do obiektu Authentication. Odnosik Login/Logout w kodzie HTML Thymeleaf może mieć postać:

```
<a sec:authorize="!isAuthenticated()" th:href="@{/login}">Login</a>
<a sec:authorize="isAuthenticated()" th:href="@{/logout}">Logout</a>
```

Login powinien już działać, natomiast Logout możemy uruchomić dołączając do łańcucha wywołań w metodzie securityFilterChain funkcję logout ustawiającą adres strony do której nastąpi przekserowanie, np. strony głównej naszej aplikacji:

```
.logout(logout -> logout
    .logoutSuccessUrl("/"))
);
```

Praktycznie cała obsługa logowania odbywa się automatycznie.

Zadania dodatkowe

1. Zdefiniuj drugiego użytkownika i pozwól mu na wyświetlanie i edycję tylko listy książek.
2. Przydziel wybrane prawa dostępu na podstawie roli użytkownika.
3. Zmień miejsce przechowywania informacji o użytkownikach z pamięci na tabelę user w bazie danych.
4. Włącz zabezpieczenia, np. CSRF, CORS i dostosuj do nich logowanie.

Opublikowanie aplikacji na serwerze foka

Wyeksportuj aplikację do pliku .war wybierając projekt i opcję Export |WAR file. Opublikuj aplikację na swoim egzemplarzu Tomcat na serwerze foka.

Sprawozdanie

W sprawozdaniu w systemie Sprawer wyślij link do działającej aplikacji wystawionej na serwerze foka oraz folder main i plik build.gradle z projektu ze źródłami spakowany do pliku .zip. W przypadku, gdy projekt nie pochodzi z Eclipse IDE, proszę dodatkowo w komentarzu napisać w jakim IDE był realizowany.