УДК 004

Управление пользователями Spring Security

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрены возможности управления пользователями в Spring Security. Описан процесс создания защиты, виды прав пользователей и их управление.
Ключевые слова: Spring Security, Java, управление

Spring Security User Management

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article has covered the user management capabilities in Spring Security. The process of creating protection, types of user rights and their management are described.
Keywords: Spring Security, Java, control

Управление пользователями в приложении определяет характеристики пользователя (учетные данные, адрес электронной почты, флаг для указания статуса учетной записи) и процесс аутентификации.

Цель данной статьи рассмотреть Spring Security и возможность управления пользователями.

С.В. Мельников провел обзор на работу с отладочным интерфейсом Java и методом модификации функциональности приложения, не изменяющий его бинарные файлы [1]. А.А.Шейн, Д.Г.Залевский, С.В. Автайкин, С.В.Карташев, С.А.Скороход разработали и описали действия программы предназначенной для автоматического создания набора классов для представления объектов модели Decode в виде нативных объектов языка

Java [2]. Н.Н. Глибовец проанализировала в своей работе особенности агентных технологий и перспективы их использования для разработки сложных многопользовательских программных систем [3]. В своей работе М.К.Ермаков, С.П.Вартанов рассмотрели вопросы проведения анализа программ интерпретируемых языков программирования [4]. Так же А.А. Птицын, Н.Л. Подколодный, Д.А.Григорович, С.В. Лаврюшев разработали молекулярно-биологический сервер и на его базе создали ряд информационно-вычислительных систем, для изучения регуляции экспрессии генов с использованием новейших технологий Java [5].

Основные компоненты управления пользователями Spring Security(рис.1):

- UserDetailsManager - этот интерфейс расширяет интерфейс UserDetailsService. Он отвечает за операции создания, обновления или удаления пользователей.
- UserDetailsService - этот интерфейс определяет метод, который находит пользователя по имени пользователя.
- UserDetails - этот интерфейс описывает пользователя.
- GrantedAuthority - пользователь может иметь один или несколько полномочий, представленных интерфейсом GrantedAuthority.

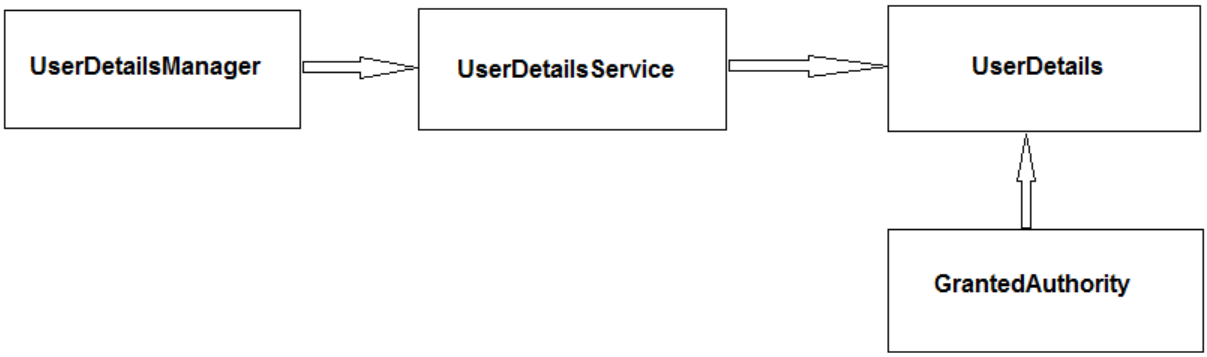

Рисунок 1 - Компоненты

Далее создадим загрузочное приложение Spring с зависимостью Spring Secutity. И определим настраиваемый класс «User» для представления пользователя, а также создадим настраиваемую службу сведений о пользователе для загрузки пользователя из списка в памяти.

Создаем загрузочное приложение Spring с зависимостью (рис.2).

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

Рисунок 2 – pom.xml

Здесь были добавлены загрузочные зависимости Spring для веб-приложений и стартера безопасности в файл «pom.xml».

Создаем собственный класс «User.java», представляющий пользователя приложения. Этот собственный пользовательский класс должен реализовать интерфейс «UserDetails» Spring Security (Рис.3).

```java
public class User implements UserDetails {

    private static final long serialVersionUID = 1L;

    private final String username;
    private final String password;
    private final String authority;

    public User(String username, String password, String authority) {
        this.username = username;
        this.password = password;
        this.authority = authority;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() { return List.of(() -> authority); }

    @Override
    public String getPassword() { return this.password; }

    @Override
    public String getUsername() { return this.username; }

    @Override
    public boolean isAccountNonExpired() { return true; }

    @Override
    public boolean isAccountNonLocked() { return true; }

    @Override
    public boolean isCredentialsNonExpired() { return true; }

    @Override
    public boolean isEnabled() { return true; }

}
```

Рисунок 3 - UserDetails

Интерфейс «UserDetails» имеет следующие методы:

- getAuthorities() - этот метод возвращает все права доступа пользователя. Расширение «GrantedAuthority» представляет собой полномочия пользователя. Эти полномочия описывают привилегии пользователя, например, чтение, запись, обновление, или действия, которые пользователь может выполнять. В данном примере есть простое строковое поле, называемое полномочиями, которое представляет полномочия пользователя. Так же возвращается «true» для следующих методов, поскольку это простая реализация.
- getPassword() и getUsername() - возвращает учетные данные пользователя (пароль и имя пользователя).
- isAccountNonExpired() - возвращает «true», если срок действия учетной записи пользователя не истек.
- isAccountNonLocked() - возвращает «true», если учетная запись пользователя не заблокирована.
- isCredentialsNonExpired() - возвращает «true», если срок действия учетных данных пользователя не истек.
- isEnabled() - возвращает «true», если пользователь включен.

Чтобы использовать только что созданный класс пользовательских сведений о пользователе, необходимо реализовать интерфейс «UserDetailsService».

«UserDetailsService» отвечает за загрузку пользователя по имени пользователя из базы данных / LDAP или любое другое хранилище (рис.4).

```
9   public class InMemoryUserDetailsService implements UserDetailsService {
10
11      private final List<UserDetails> users;
12
13      public InMemoryUserDetailsService(List<UserDetails> users) { this.users = users; }
16
17      @Override
18      public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
19          return users.stream()
20                  .filter(u -> u.getUsername().equals(username))
21                  .findFirst()
22                  .orElseThrow(() -> new UsernameNotFoundException("Пользователь не найден"));
23      }
24  }
```

Рисунок 4 - InMemoryUserDetailsService

Здесь идет поддержка пользователей в локальной коллекции под названием «users». Метод «loadUserByUsername» ищет доступные пользовательские экземпляры из коллекции «users» и возвращает пользовательский экземпляр, если имя пользователя совпадает. Если имя пользователя не соответствует ни одному из существующих пользователей, генерируется исключение «UsernameNotFoundException».

Чтобы связать пользовательскую реализацию «UserDetails» и «UserDetailsService» с безопасностью Spring, необходимо настроить bean-компонент «UserDetailsService» (рис.5).

```
@Configuration
public class AuthConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails u = new User( username: "Pavel",  password: "12345",  authority: "READ");
        List<UserDetails> users = List.of(u);
        return new InMemoryUserDetailsService(users);
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return NoOpPasswordEncoder.getInstance(); }
}
```

Рисунок 5 - UserDetailsService

Был создан экземпляр класса «User», который является настроенным экземпляром «UserDetails». У пользователя будет имя «Pavel», пароль «12345» и полномочия «READ». Также был создан список пользователей и передан в конструктор класса «InMemoryUserDetailsService», который реализует интерфейс «UserDetailsSerice». Коллекция пользователей, переданная конструктору, добавляет пользователей в локально определенный список пользователей в памяти класса «InMemoryUserDetailsService». Необходимо также определить bean-компонент кодировщика паролей, если нужно настраивать «UserDetailsService».

Далее создаем простую конечную точку HTTP GET, пользователь попадет на этот путь после успешной аутентификации (рис.6).

```
@RestController
public class HelloRestController {

    @GetMapping("/")
    public String getMessage() { return "Hello!!"; }
}
```

Рисунок 6 - HelloRestController

Осталось проверить работу проекта. Запускаем класс загрузки Spring и переходим в браузер по локальному адресу. Spring Security отображает страницу входа по умолчанию. Вводим учетные данные пользователя и нажимаем кнопку «Sing in» (рис.7).

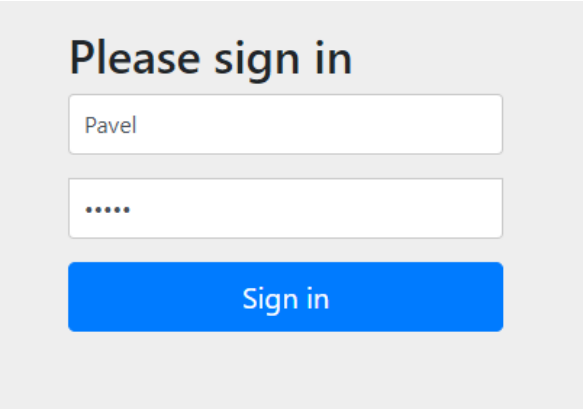

Рисунок 7 – окно авторизации

После успешной авторизации, пользователя перенаправляет на страницу с приветствием (рис.8).

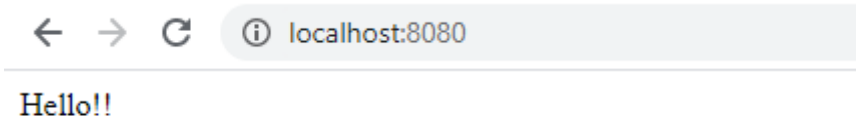

Рисунок 8 – Окно приветствия

В данной статье были рассмотрены основы безопасности Spring, как Spring Security предоставляет данные пользователя, полномочия, права. Рассмотрели базовые функции, необходимые для загрузки, управления пользователями и их аутентификации.

Библиографический список

1. Шейн А.А., Залевский Д.Г., Автайкин С.В., Карташев С.В., Скороход С.А. Генератор исходного кода на языке java по описанию бортовых компонентов decode (decode java generator 0.2) // Вестник Волжского Университета им. В.Н. Татищева. 2019. №3. С. 26-32.
2. Мельников С.В. Обзор и применение отладочного интерфейса java (jdi) для обратимой модификации программных продуктов // Современные проблемы науки и образования. 2018. №8. С. 8-19.
3. Глибовец Н.Н. Использование jade (java agent development environment) для разработки компьютерных систем поддержки дистанционного обучения агентного типа // Заметки по информатике и математике. 2019. №10. С. 15-20.
4. Ермаков М.К., Вартанов С.П. Подход к проведению динамического анализа java-программ методом модификации виртуальной машины java // Научные труды Винницкого национального технологического университета. 2018. №6. С. 10-17.
5. Птицын А.А., Подколодный Н.Л., Григорович Д.А., Лаврюшев С.В. Создание молекулярно-биологического сервера www с использованием новейших технологий java // Заметки по информатике и математике. 2020. №1. С. 11-20.