УДК 004

Настройка сервера администратора и клиента с использованием Spring Boot

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается процесс настройки серверной части администратора для мониторинга подключившихся клиентов. Использоваться будет Spring Boot server admin. Практическим результатом является настроенный сервер администратора.
Ключевые слова: Spring Boot, Java, Admin

Configuring admin server and client using Spring Boot

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of configuring the admin backend to monitor connected clients. Spring Boot server admin will be used. The bottom line is a customized admin server.
Keywords: Spring Boot, Java, Admin

Spring Boot admin server - это проект сообщества, который значительно упрощает мониторинг приложений загрузки Spring. Клиентское приложение может быть зарегистрировано на сервере администратора через HTTP или обнаружено с помощью Spring Cloud.

Цель данной работы настроить сервер администрирования Spring Boot и зарегистрировать на нем клиентское приложение.

В своей работе А.Б.Джемалетдинов, А.А.Шевченко рассмотрели вопросы создания тестов для Spring Boot mvc контроллеров [1]. В.И.Зарайский провел обзор на разработку модуля автоматизации работы с

конференциями в кафедральном приложении [2]. Р.И.Ибраимов продемонстрировал процесс создания Docker-образа для Spring Boot проекта и развернул его на платформе AWS EC2[3]. Е.О.Кабардинский, А.Г.Ивашко провели сравнительный анализ сервисных шин предприятия, а так же сравнили некоторые ESB, одна из которых Spring Boot [4]. Так же Р.И.Ибраимов, А.Р.Зайчик, Н.С.Минзатров разработали генеалогическое дерево на языке Java с использованием фреймворка Spring Boot b ,b,kbjntrb gedcom4j[5].

Создадим новое приложение Spring с необходимыми зависимостями. Добавим «spring-boot-starter-web», «spring-boot-starter-security» и «spring-boot-admin-starter-server».

«spring-boot-admin-starter-server» зависимость, которая используются для того, чтобы приложение работало в качестве сервера администратора Spring Boot (рис.1).

```
<dependencies>
    <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>de.codecentric</groupId>
        <artifactId>spring-boot-admin-starter-server</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>de.codecentric</groupId>
            <artifactId>spring-boot-admin-dependencies</artifactId>
            <version>${spring-boot-admin.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
```

Рисунок 1 – pom.xml

Чтобы включить возможности административного сервера для созданного приложения, необходимо аннотировать приложение аннотацией «@EnableAdminServer» (рис.2).

```
package com.asbnotebook;

import ...

@SpringBootApplication
@EnableAdminServer
public class SpringBootAdminServerExampleApplication {

    public static void main(String[] args) { SpringApplication.run(SpringBootAdminServerExampleApplication.class, args); }
}
```

Рисунок 2 - SpringBootAdminServerExampleApplication

Был изменен порт сервера на 9090, а также установлено имя пользователя и пароль по умолчанию (рис.3).

```
server.port=9090
spring.security.user.name=admin
spring.security.user.password=admin
```

Рисунок 3 - application.properties

Создадим класс конфигурации безопасности и добавим следующие детали конфигурации Spring (рис.4).

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessHandler;
import org.springframework.security.web.csrf.CookieCsrfTokenRepository;

@Configuration(proxyBeanMethods = false)
public class SecuritySecureConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        SavedRequestAwareAuthenticationSuccessHandler successHandler = new SavedRequestAwareAuthenticationSuccessHandler();
        successHandler.setTargetUrlParameter("redirectTo");
        successHandler.setDefaultTargetUrl("/");

        http.authorizeRequests(authorizeRequests -> authorizeRequests.antMatchers( ...antPatterns: "/assets/**").permitAll()
                .antMatchers( ...antPatterns: "/login").permitAll().anyRequest().authenticated())
                .formLogin().loginPage("/login").and()
                .logout().logoutUrl("/logout").and()
                .httpBasic(Customizer.withDefaults())
                .csrf().csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
                .ignoringAntMatchers("/instances", "/actuator/**");
    }
}
```

Рисунок 4 - SecuritySecureConfig

Этот класс настраивает базовую конфигурацию Spring Security для административного сервера Spring Boot.

Создадим еще одно приложение Spring Boot с необходимыми зависимостями. Оно будет клиентским приложением, которое будет зарегистрировано на сервере администратора.

Добавим зависимости «spring-boot-starter-web», «spring-boot-starter-actator» и «spring-boot-admin-starter-client» (рис.5).

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>de.codecentric</groupId>
        <artifactId>spring-boot-admin-starter-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

Рисунок 5 – Pom.xml

Чтобы зарегистрировать клиентское приложение на сервере администратора, необходимо добавить следующие конфигурации (рис.6):

- «spring.boot.admin.client.url» - URL-адрес сервера администратора.
- «spring.boot.admin.client.username» - Имя пользователя административного сервера.
- «spring.boot.admin.client.password» - Пароль административного сервера.
- «management.endpoints.web.exposure.include» - чтобы открыть все конечные точки привода.
- «spring.jmx.enabled» - открыть компоненты управления JMX.

```
spring.boot.admin.client.url=http://localhost:9090
spring.boot.admin.client.username=admin
spring.boot.admin.client.password=admin

management.endpoints.web.exposure.include=*
spring.jmx.enabled=true
```

Рисунок 6 - application.properties

Создадим простую конечную точку GET, создав класс «RestController» (рис.7).

```
import ...

@RestController
public class TestController {

    @GetMapping("/test-admin")
    public String testMethod() { return "Тестовая страница"; }
}
```

Рисунок 7 - TestController

Запустим оба приложения (сервер и клиент). Можно получить доступ к консоли администратора сервера Spring по адресу «http://localhost:9090» .

Поскольку была включена конфигурация безопасности Spring, то при входе получим страницу входа на сервер администратора по умолчанию (Рис.8).

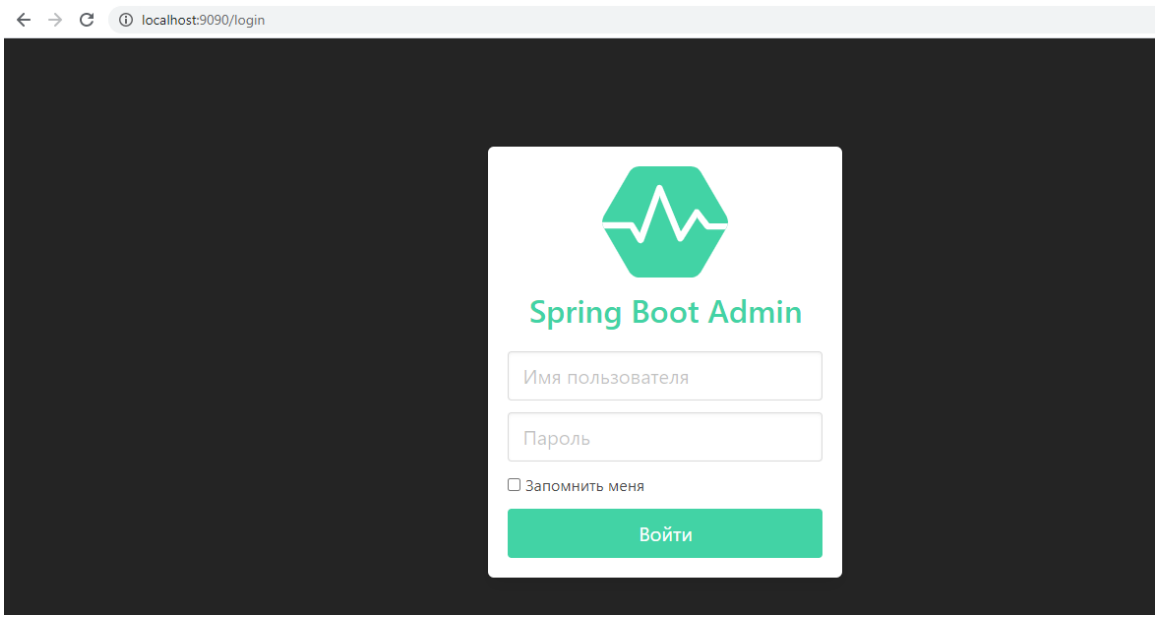

Рисунок 8 – Панель авторизации администратора

После входа в консоль сервера администратора можно наблюдать за зарегистрированными клиентскими приложениями и экземплярами (Рис.9).

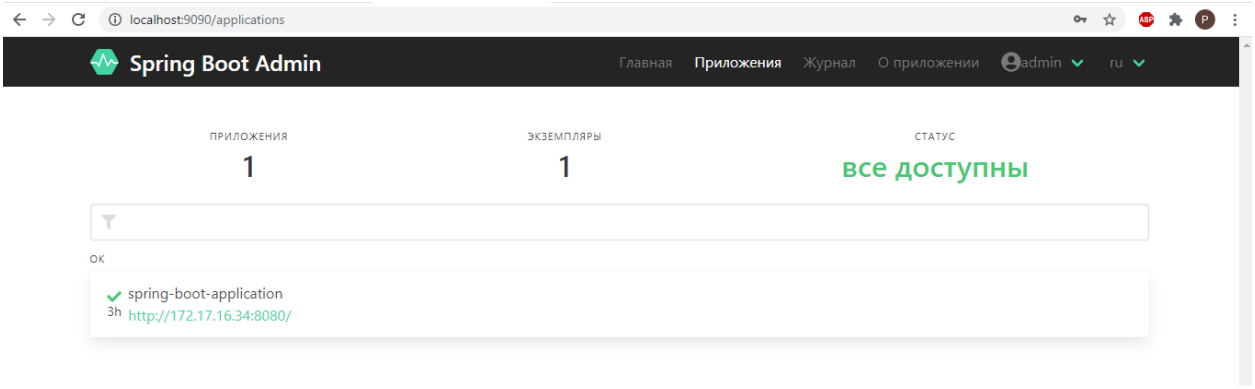

Рисунок 9 – Панель администратора

В разделе показателей отображаются сведения о зарегистрированной метрики приложения (рис.10).

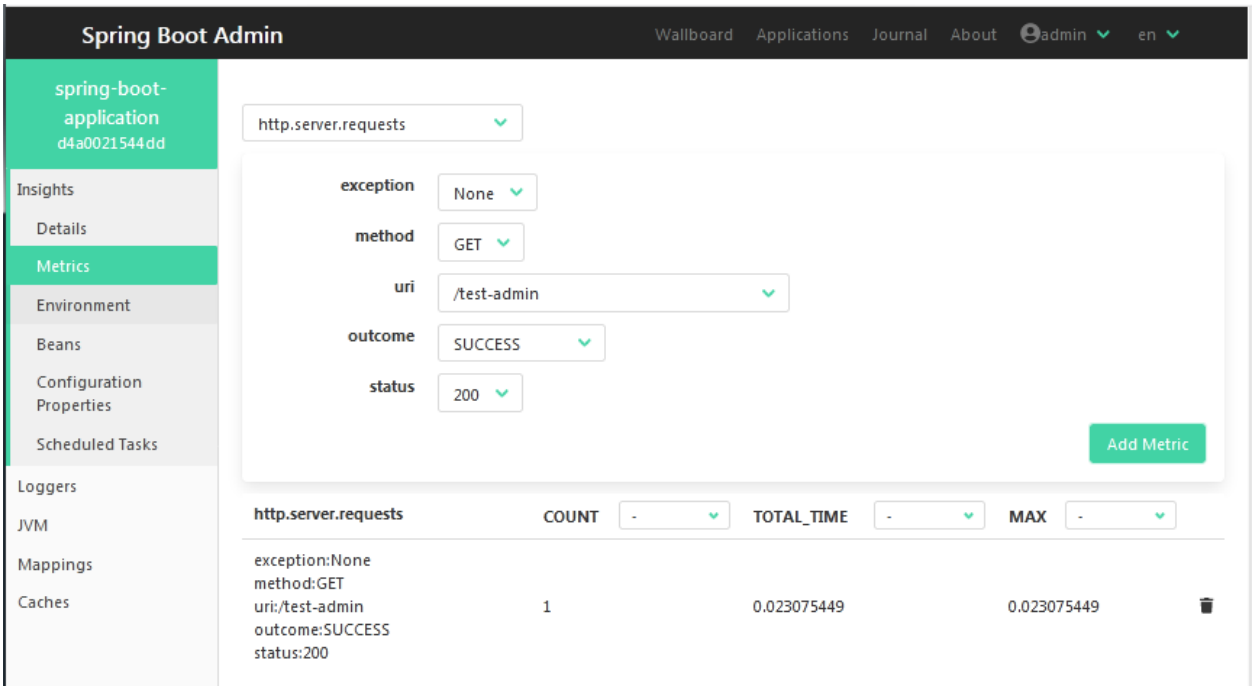

Рисунок 10 – Панель метрики

При попытки запустить клиентское приложение, созданное для проведения тестирования, будет сделан переход на такую страницу (рис.11).

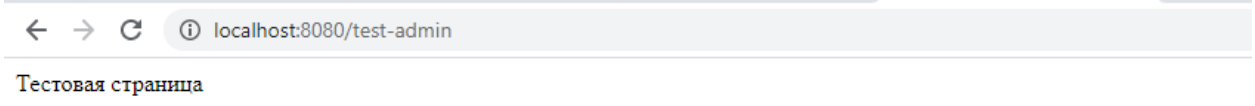

Рисунок 11 – Окно пользователя

В данной статье были изучены основы по настройке административного сервера Spring Boot, создан сам сервер администратора, настроили безопасность с помощью Spring Security, а так же создано клиентское приложение для проведения тестирования.

Библиографический список

1. Джемалетдинов А.Б., Шевченко А.А. Spring boot: создание тестов для spring mvc контроллеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №4(18). С. 104-111.
2. Зарайский В.И. Разработка модуля автоматизации работы с конференциями в кафедральном приложении // Вестник ульяновского государственного технического университета. 2019. №3. С. 74-82.
3. Ибраимов Р.И. Развертывание spring приложения с помощью сервиса aws ec2 и docker-контейнеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2020. №1(27). С. 138-147.
4. Кабардинский Е.О., Ивашко А.Г. Сравнительный анализ сервисных шин предприятия (esb) // Математическое и информационное моделирование. 2017. №10. С. 177-185.
5. Ибраимов Р.И., Зайчик А.Р., Минзатров Н.С. Разработка генеалогического дерева средствами фреймвока spring boot // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №4(18). С. 18-23.