УДК 004

Создание графиков с помощью 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 рассмотрены возможности интеграции библиотеки JS Highcharts в приложение Spring. Разработано вэб – приложение, которое считывает данные JSON через вызов AJAX и отображает графики на странице HTML.
Ключевые слова: Spring Boot, Java, Highcharts

Generating graphs with Spring Boot

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article explored the possibilities of integrating the Highcharts JS library into a Spring application. Developed a web application that reads JSON data through an AJAX call and displays graphs on an HTML page.
Keywords: Spring Boot, Java, Highcharts

Highcharts - одна из популярных библиотек javascript, используемых для создания удобных диаграмм для заданного набора данных. С ее помощью можно генерировать различные типы графических представлений данных с помощью.

Цель данной статьи написать вэб-приложение, которое будет считывать данные JSON через вызов AJAX.

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].

Для начала создаем приложение Spring и добавляем зависимости «spring-boot-starter-web» и «spring-boot-starter-thymeleaf» (рис.1)

```
<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>
    <exclusions>
        <exclusion>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
    <version>2.1.5.RELEASE</version>
    <scope>compile</scope>
</dependency>
```

Рисунок 1 – pom.xml

Чтобы перенаправить веб-приложение Spring на целевую страницу по умолчанию, внесем изменения в класс загрузочного приложения Spring, где реализуем интерфейс «WebMvcConfigurer» и переопределяем метод «addViewControllers», чтобы установить целевое представление по умолчанию на «landig-page» (рис.2).

```java
@SpringBootApplication
public class SpringBootHighchartsExampleApplication implements WebMvcConfigurer {

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

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController( urlPath: "/").setViewName("landing-page");
    }

}
```

Рисунок 2 – SpringBootHighchartsExampleApplication

Далее нужно создать конечную точку «RESTful» для возврата необходимых данных. Для этого создаем контроллер «REST» с именем «HighChartController.java»(рис.3).

```java
@RestController
public class HighChartController {

    @GetMapping("/get-data")
    public ResponseEntity<Map<String, Integer>> getPieChart() {
        Map<String, Integer> graphData = new TreeMap<>();
        graphData.put("2016", 147);
        graphData.put("2017", 1256);
        graphData.put("2018", 3856);
        graphData.put("2019", 19807);
        return new ResponseEntity<>(graphData, HttpStatus.OK);
    }
}
```

Рисунок 3 – HighChartController

Здесь возвращается карта со статистикой. Ключи карты имеют календарные годы, а значения карты - количество просмотров страниц. Конечная точка «RESTful» отображается в пути «/get-data». В данной работе используются статические данные, возвращая жестко закодированные значения. Также можно вернуть данные из базы данных, используя Spring data jpa и другие способы.

Ранее уже была сопоставлена целевая страница, поэтому установим ее имя как на предыдущем шаге. Создаем HTML страницу под названием «landig-page.html» (рис.4-6).

```
<head>
    <meta charset="ISO-8859-1">
    <title>High Chart Example - Spring Boot</title>
    <link rel="stylesheet"
          href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <link rel="stylesheet"
          href="https://code.highcharts.com/css/highcharts.css" />
    <script
            src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script
            src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
    <script src="https://code.highcharts.com/highcharts.js"></script>
    <script src="https://code.highcharts.com/modules/exporting.js"></script>
    <script src="https://code.highcharts.com/modules/export-data.js"></script>
    <script src="https://code.highcharts.com/modules/accessibility.js"></script>
</head>
<body>
<div align="center">
    <h2>Spring Boot Highcharts Example</h2>
</div>
<figure class="highcharts-figure">
    <div id="container-bar"></div>
</figure>
```

Рисунок 4 - landing-page.html

```
<script type="text/javascript">
    $(document).ready(
            function() {
                $.ajax({
                    url : "/get-data",
                    success : function(result) {
                        var yearDtls = [];
                        var countDtls = [];
                        console.log(result);
                        Object.keys(result).forEach(
                                function(key) {
                                    yearDtls.push(key);
                                    countDtls.push(result[key]);
                                });
                        drawChart(yearDtls, countDtls);
                    }
                });
            });
```

Рисунок 5 - landing-page.html

```
    function drawChart(year, viewCounts) {
        Highcharts.chart('container-bar', {
            chart : {
                type : 'column',
                styledMode : true
            },
            title : {
                text : 'Blog Page view count'
            },
            xAxis : [ {
                title : {
                    text : 'Year'
                },
                categories : year
            } ],
            yAxis : [ {
                className : 'highcharts-color-0',
                title : {
                    text : 'Page View Count'
                }
            } ],
            series : [ {
                data : viewCounts
            } ]
        });
    }
</script>
</body>
```

Рисунок 6 - landing-page.html

Здесь добавляются необходимые библиотеки javascript и файлы CSS. Также добавили файлы CSS и js для «Bootstrap» и «Highchart», и добавили библиотеку javascript «JQuery» в часть заголовка HTML-страницы.

Далее загружаем данные JSON из бэкэнда, чтобы показать гистограмму через вызов «JQuery AJAX» в конечную точку «/get-data». После загрузки данных идет постройка диаграммы с помощью метода «Highcahrts.chart()» библиотеки «Highcharts». Этот метод принимает идентификатор HTML-элемента и конфигурацию диаграммы в качестве параметра метода. Передаем «container-bar» в качестве идентификатора HTML-элемента

Некоторые из конфигураций Highchart

Chart - настройка, относящаяся к диаграмме.

Title - для установки заголовка Highchart.

xAxis - для настройки конфигурации, связанной с осью X графика.

yAxis - для настройки конфигурации, связанной с осью Y графика.

Series - список значений, которые будут использоваться для построения диаграммы.

Теперь запускаем приложение и получаем графики (рис.7).

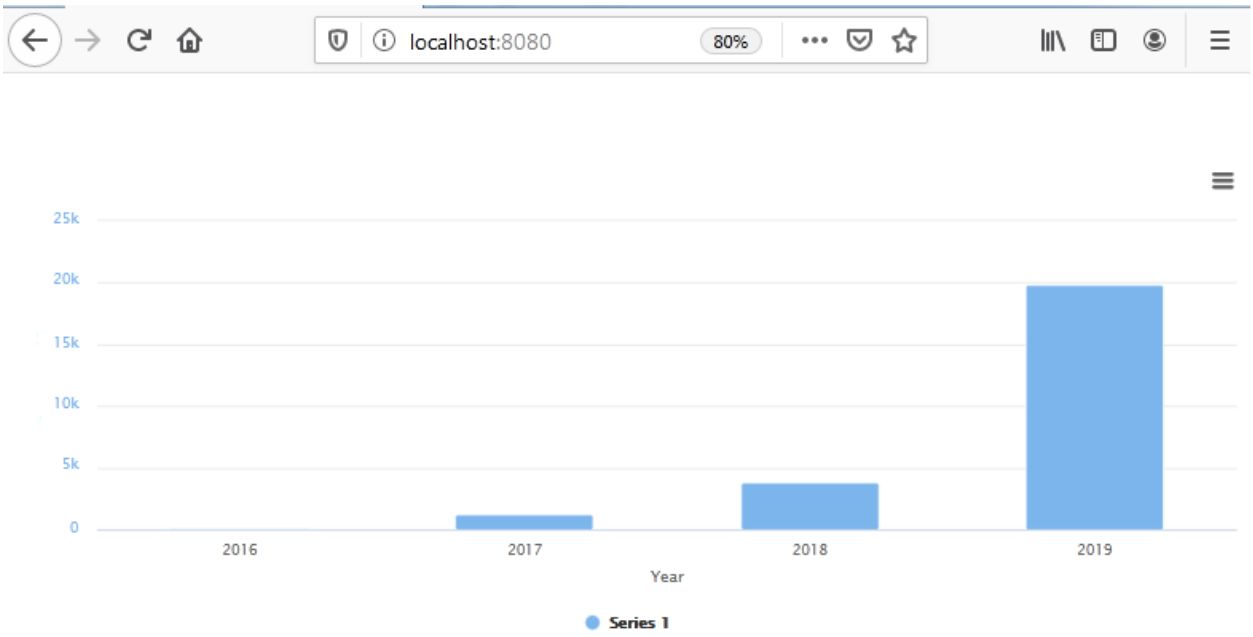

Рисунок 7 – Пример отображения

В данной статье была рассмотрена возможность интеграции библиотеки Highcharts в приложение Spring Boot, а так же была изучена возможность загрузки данных на html страницу с помощью вызова AJAX.

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.