УДК 004

Отправка электронного письма из приложения SpingBoot

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрена возможность отправки электронных сообщений. Так же описаны методы добавления вложенных файлов, таких как фотография и файл pdf.
Ключевые слова: e-mail, Java, сообщения

Sending an Email from the SpingBoot App

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the ability to send emails. Methods for adding attachments such as photo and pdf file are also described.
Keywords: e-mail, Java, messages

Отправка сообщений на электронные адреса с помощью скриптов помогает компаниям уменьшить человеческие ресурсы. Так же можно с легкостью сделать массовую рассылку, добавив всего несколько значений.

Цель данной статьи рассмотреть возможности Spring Boot и отправить сообщение из приложения.

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 [1]. Н.Н. Глибовец проанализировала в своей работе особенности агентных технологий и перспективы их использования для разработки сложных многопользовательских программных систем [2]. В своей работе

М.К.Ермаков, С.П.Вартанов рассмотрели вопросы проведения анализа программ интерпретируемых языков программирования [3]. С.В. Мельников провел обзор на работу с отладочным интерфейсом Java и методом модификации функциональности приложения, не изменяющий его бинарные файлы [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].

В качестве тела письма будем использовать шаблон «FreeMarker». «FreeMarker» - это шаблонизатор, который генерирует динамические HTML-файлы путем обработки значений, переданных с уровня приложения.

Сперва создадим загрузочное приложение «Spring» с необходимыми зависимостями. Добавим зависимости: spring-boot-starter-web, spring-boot-starter-freemarker, lombok и spring-boot-starter-mail. Все это добавляется в файл конфигурации «pom.xml»(рис.1-2).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
        <relativePath/>
    </parent>
    <groupId>com.asb.example</groupId>
    <artifactId>spring-boot-mailing-example</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-boot-mailing-example</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-freemarker</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
```

Рисунок 1 – pom.xml

```
        <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>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Рисунок 2 – pom.xml

Далее необходимо добавить указанные свойства конфигурации почты в файл конфигурации «application.properties» (рис.3).

```
spring.mail.default-encoding=UTF-8
spring.mail.host=smtp.gmail.com
spring.mail.protocol=smtp
spring.mail.username=pasandreg@gmail.com
spring.mail.password=Пароль от почты
spring.mail.port=587
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Рисунок 3 - application.properties

Некоторые из используемых свойств конфигурации:

- spring.mail.protocol: Протокол, используемый для отправки почты.
- Spring.mail.host: в этом примере используется Gmail в качестве хоста SMTP.
- spring.mail.username: имя пользователя для входа на SMTP-сервер. Здесь необходимо ввести логин почты.
- spring.mail.password: пароль SMTP-сервера. Здесь необходимо ввести пароль Gmail.

- spring.mail.properties : другие свойства почты Java.

Далее создаем файл конфигурации «EmailConfig.java» и определяем компонент конфигурации шаблона FreeMarker. Устанавливаем путь к файлу шаблона. Этот файл шаблона используется в качестве тела письма (рис.4).

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.ui.freemarker.FreeMarkerConfigurationFactoryBean;

@Configuration
public class EmailConfig {

    @Primary
    @Bean
    public FreeMarkerConfigurationFactoryBean fmfbean() {
        FreeMarkerConfigurationFactoryBean bean = new FreeMarkerConfigurationFactoryBean();
        bean.setTemplateLoaderPath("classpath:/templates");
        return bean;
    }
}
```

Рисунок 4 - EmailConfig.java

Так же создаем DTO-класс «EmailRequestDto.java». Этот класс используется в качестве объекта запроса для отправки электронного письма (рис.5).

```java
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class EmailRequestDto {

    private String from;
    private String to;
    private String subject;
    private String name;

}
```

Рисунок 5 - EmailRequestDto.java

Следующим файлом создаем класс службы Spring с именем «MailService.java». Этот класс обработает файл шаблона FreeMarker, создаст тело письма и отправит запрос на SMTP-сервер (рис.6).

```java
@Service
public class MailService {

    @Autowired
    private JavaMailSender mailSender;

    @Autowired
    private Configuration configuration;

    public String sendMail(EmailRequestDto request, Map<String, String> model) {

        String response;
        MimeMessage message = mailSender.createMimeMessage();
        try {
            MimeMessageHelper helper = new MimeMessageHelper(message, MimeMessageHelper.MULTIPART_MODE_MIXED_RELATED,
                    StandardCharsets.UTF_8.name());

            ClassPathResource pdf = new ClassPathResource("static/test.pdf");
            ClassPathResource image = new ClassPathResource("static/test.png");
            Template template = configuration.getTemplate( name: "email.ftl");
            String html = FreeMarkerTemplateUtils.processTemplateIntoString(template, model);

            helper.setTo(request.getTo());
            helper.setFrom(request.getFrom());
            helper.setSubject(request.getSubject());
            helper.setText(html,  html: true);
            helper.addInline( contentId: "test", image);
            helper.addAttachment( attachmentFilename: "test.pdf", pdf);

            mailSender.send(message);
            response = "Успешно отправлено пользователю :" + request.getTo();
        } catch (MessagingException | IOException | TemplateException e) {
            response = "Сообщение не отправлено пользователю :" + request.getTo();
        }
        return response;
    }
}
```

Рисунок 6 - MailService.java

Здесь есть метод «sendMail()», который принимает запросы электронной почты и мапинг в качестве входного параметра. Spring предоставляет класс «JavaMailSender», который можно использовать для отправки электронной почты. Создан экземпляр «MimeMessage» и экземпляр «MimeMessageHelper». Этот вспомогательный класс можно использовать для установки различных свойств электронной почты, таких как отправка, текст сообщения электронной почты, вложения и т.д. Так же использован метод «addInline()» для вставки изображения в тело письма. Был прикреплен PDF - файл с помощью «addAttachment ()» метод «MimeMessageHelper».

Шаблон FreeMarker обрабатывается и преобразуется в текст HTML с помощью экземпляра «Configuration». Класс «FreeMarkerTemplateUtils» помогает динамически обрабатывать шаблон FreeMarker и генерирует окончательный формат HTML. Эти два класса доступны в библиотеке «spring-boot-starter-freemarker».

Далее создаем класс «MailSenderController.java». Это контроллер «RESTful» с одним методом «POST», который отправляет электронное письмо по указанному идентификатору электронной почты с темой (рис.7).

```java
import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.asb.example.dto.EmailRequestDto;
import com.asb.example.service.MailService;

@RestController
public class MailSenderController {

    @Autowired
    private MailService mailService;

    @PostMapping("send-mail")
    public ResponseEntity<String> sendMail(@RequestBody EmailRequestDto emailRequest) {
        Map<String, String> model = new HashMap<>();
        model.put("name", emailRequest.getName());
        model.put("value", "Тема Письма");
        String response = mailService.sendMail(emailRequest, model);
        return new ResponseEntity<>(response, HttpStatus.OK);
    }
}
```

Рисунок 7 - MailSenderController.java

И последнее создаем файл шаблона Freemarker под названием «email.ftl» под «resources/template/ directory». Этот файл шаблона содержит заголовок, приветствие с именем получателя почты, приветственное сообщение и изображение в теле письма (рис.8).

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Отправка сообщения из приложения Spring</title>
  </head>
  <body>
    <h2>Сообщение отправлено из Spring</h2>
    <h3>Hi ${name}!!</h3>
    <p>${value}</p>
    <div>
        <a href="www.youtube.com">
            <img src='cid:test' alt="test" style="width:10%"/>
            <p>ссылка на YouTube</p>
        </a>
    </div>
  </body>
</html>
```

Рисунок 8 - email.ftl

Теперь осталось протестировать приложение. Отправим POST запрос с помощью «postman» (рис.10).

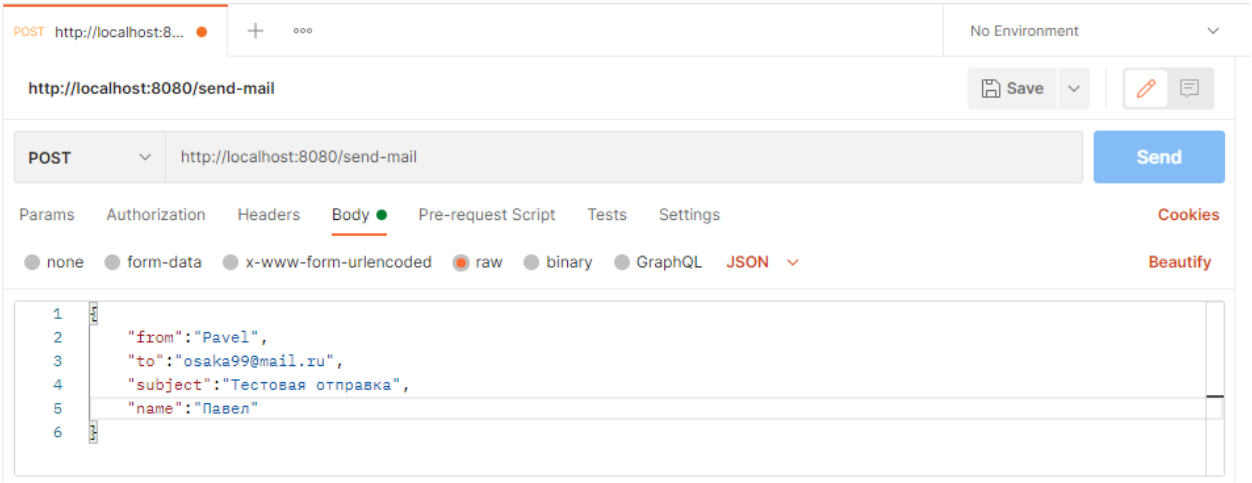

Рисунок 10 – POST запрос

После отправки, приложение получает данные для отправки и отсылает письмо (рис.11).

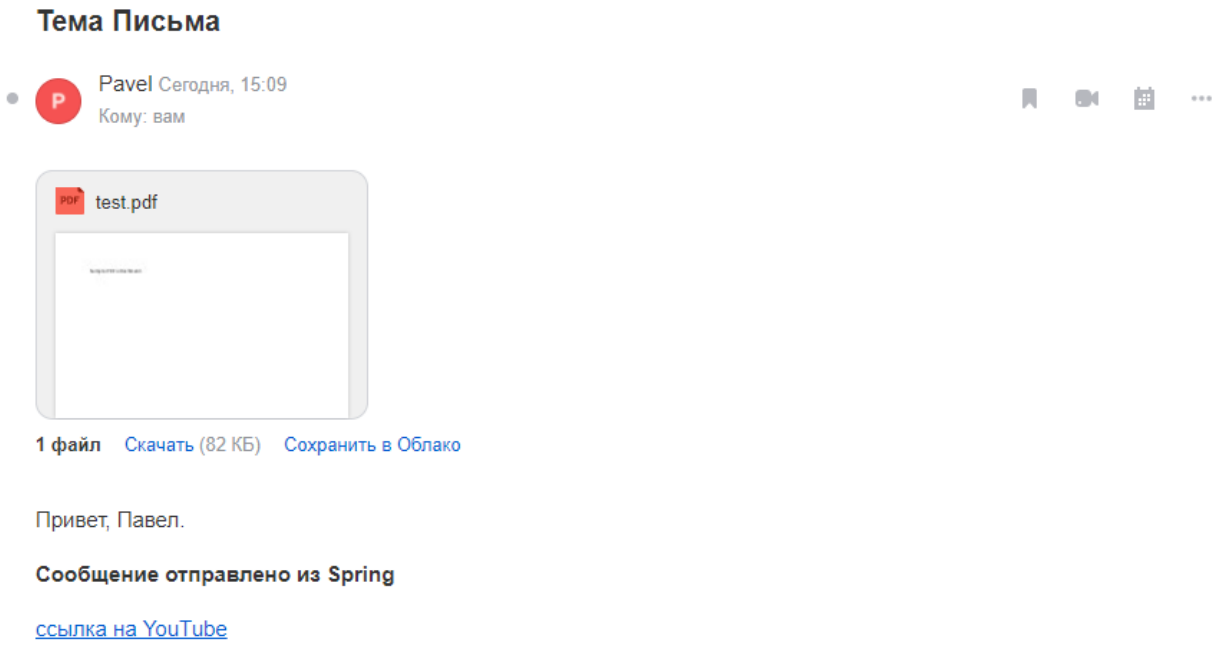

Рисунок 11 – Получение письма

В этой статье была рассмотрена возможность отправки электронного письма с помощью загрузочного приложения Spring. Также рассмотрели, как использовать шаблон FreeMarker в качестве тела письма и встраивать файл в письмо.

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. 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.
3. 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.
4. 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.
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.