使用教程

一、开启邮件服务

二、配置邮件服务

通过 Spring Initializr 创建工程springboot-send-mail;
然后在pom.xml 引入web、thymeleaf 和spring-boot-starter-mail等相关依赖。

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

```
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>webjars-locator-core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>jquery</artifactId>
            <version>3.3.1</version>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>bootstrap</artifactId>
            <version>3.3.7</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

根据前面提到的配置项(MailProperties)填写相关配置信息，其中spring.mail.username 表示连接邮件服务器时认证的登陆账号，可以是普通的手机号或者登陆账号，并非一定是邮箱，为了解决这个问题，推荐大家在spring.mail. properties.from填写邮件发信人即真实邮箱。

然后在application.yml添加如下配置：

```
spring:
  mail:
    host: smtp.163.com #SMTP服务器地址
    username: socks #登陆账号
    password: 123456 #登陆密码（或授权码）
    properties:
      from: socks@163.com #邮件发信人（即真实邮箱）
  thymeleaf:
    cache: false
    prefix: classpath:/views/
  servlet:
    multipart:
      max-file-size: 10MB #限制单个文件大小
      max-request-size: 50MB #限制请求总量
```

在发送邮件前，需要先构建 SimpleMailMessage或 MimeMessage 邮件信息类来填写邮件标题、邮件内容等信息，最后提交给JavaMailSenderImpl发送邮件，这样看起来没什么问题，也能实现既定

目标，但在实际使用中会出现大量零散和重复的代码，还不便于保存邮件到数据库。

邮件信息类(MailVo) 来保存发送邮件时的邮件主题、邮件内容等信息

```
public class MailVo {
    /**
     * 邮件id
     */
    private String id;
    /**
     * 邮件发送人
     */
    private String from;
    /**
     * 邮件接收人（多个邮箱则用逗号","隔开）
     */
    private String to;
    /**
     * 邮件主题
     */
    private String subject;
    /**
     * 邮件内容
     */
    private String text;
    /**
     * 发送时间
     */
    private Date sentDate;
    /**
     * 抄送（多个邮箱则用逗号","隔开）
     */
    private String cc;
    /**
     * 密送（多个邮箱则用逗号","隔开）
     */
    private String bcc;
    /**
     * 状态
     */
    private String status;
    /**
     * /报错信息
     */
    private String error;
    /**
     * 邮件附件
     */
    @JsonIgnore
    private MultipartFile[] multipartFiles;
```

三、发送邮件和附件

除了发送邮件之外，还包括检测邮件和保存邮件等操作

- 检测邮件 checkMail(); 首先校验邮件收信人、邮件主题和邮件内容这些必填项，若为空则拒绝发送。
- 发送邮件 sendMimeMail(); 其次通过MimeMessageHelper来解析MailVo并构建MimeMessage传输邮件。
- 保存邮件 sendMimeMail(); 最后将邮件保存到数据库，便于统计和追查邮件问题。

MailService服务类

```
package com.legend.springboot.service;

import com.legend.springboot.vo.MailVo;
import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import javax.mail.MessagingException;

/**
 * 邮件业务类 MailService
 *
 * @author legend
 */
@Service
public class MailService {
    /**
     * 提供日志类
     */
    private Logger logger = LoggerFactory.getLogger(getClass());

    /**
     * 注入邮件工具类
     */
    @Autowired
    private JavaMailSenderImpl mailSender;

    /**
     * 发送邮件
     *
     * @param mailVo
     * @return
```

```
     */
    public MailVo sendMail(MailVo mailVo) {
        try {
            //1.检测邮件
            checkMail(mailVo);
            //2.发送邮件
            sendMimeMail(mailVo);
            //3.保存邮件
            return saveMail(mailVo);
        } catch (Exception e) {
            //打印日志
            logger.error("发送邮件失败:", e);
            mailVo.setStatus("fail");
            mailVo.setError(e.getMessage());
            return mailVo;
        }
    }

    /**
     * 检测邮件信息类
     *
     * @param mailVo
     */
    private void checkMail(MailVo mailVo) {
        if (StringUtils.isEmpty(mailVo.getTo())) {
            throw new RuntimeException("邮件收信人不能为空");
        }
        if (StringUtils.isEmpty(mailVo.getSubject())) {
            throw new RuntimeException("邮件主题不能为空");
        }
        if (StringUtils.isEmpty(mailVo.getText())) {
            throw new RuntimeException("邮件内容不能为空");
        }
    }

    /**
     * 构建复杂邮件信息类
     *
     * @param mailVo
     */
    private void sendMimeMail(MailVo mailVo) {
        try {
            //true表示支持复杂类型
            MimeMessageHelper messageHelper = new MimeMessageHelper(mailS
ender.createMimeMessage(), true);
            //邮件发信人从配置项读取
            mailVo.setFrom(getMailSendFrom());
            messageHelper.setFrom(mailVo.getFrom());
            //邮件发信人
            messageHelper.setFrom(mailVo.getFrom());
            //邮件收信人
            messageHelper.setTo(mailVo.getTo().split(","));
```

```
            //邮件主题
            messageHelper.setSubject(mailVo.getSubject());
            //邮件内容
            messageHelper.setText(mailVo.getText());
            //抄送
            if (!StringUtils.isEmpty(mailVo.getCc())) {
                messageHelper.setCc(mailVo.getCc().split(","));
            }
            //密送
            if (!StringUtils.isEmpty(mailVo.getBcc())) {
                messageHelper.setCc(mailVo.getBcc().split(","));
            }
            //添加邮件附件
            if (mailVo.getMultipartFiles() != null) {
                for (MultipartFile multipartFile : mailVo.getMultipartFil
es()) {
                    messageHelper.addAttachment(multipartFile.getOriginal
Filename(), multipartFile);
                }
            }
            //发送时间
            /*if (StringUtils.isEmpty(mailVo.getSentDate())) {
                mailVo.setSentDate(new Date());
                messageHelper.setSentDate(mailVo.getSentDate());
            }*/
            //正式发送邮件
            mailSender.send(messageHelper.getMimeMessage());
            mailVo.setStatus("ok");
            logger.info("发送邮件成功：{}->{}", mailVo.getFrom(), mailVo.get
To());
        } catch (MessagingException e) {
            //发送失败
            throw new RuntimeException(e);
        }
    }

    /**
     * 保存邮件
     *
     * @param mailVo
     * @return
     */
    private MailVo saveMail(MailVo mailVo) {
        //TODO 将邮件保存到数据库..
        return mailVo;
    }

    /**
     * 获取邮件发信人
     *
     * @return
     */
```

```
    public String getMailSendFrom() {
        return mailSender.getJavaMailProperties().getProperty("from");
    }
}
```

MainController控制器

```
package com.legend.springboot.controller;

import com.legend.springboot.service.MailService;
import com.legend.springboot.vo.MailVo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

/**
 * 邮箱控制器
 *
 * @author legend
 */
@RestController
public class MailController {

    private final MailService mailService;

    @Autowired
    public MailController(MailService mailService) {
        this.mailService = mailService;
    }

    /**
     * 发送邮件的主界面
     */
    @GetMapping("/")
    public ModelAndView index() {
        //打开发送邮件的页面
        ModelAndView mv = new ModelAndView("sendMail");
        //邮件发信人
        mv.addObject("from", mailService.getMailSendFrom());
        return mv;
    }

    /**
     * 发送邮件
     */
    @PostMapping("/mail/send")
```

```
    public MailVo sendMail(MailVo mailVo, MultipartFile[] files) {
        mailVo.setMultipartFiles(files);
        //发送邮件和附件
        return mailService.sendMail(mailVo);
    }
}
```

然后在/resources/templates 目录新建sendMail.html，

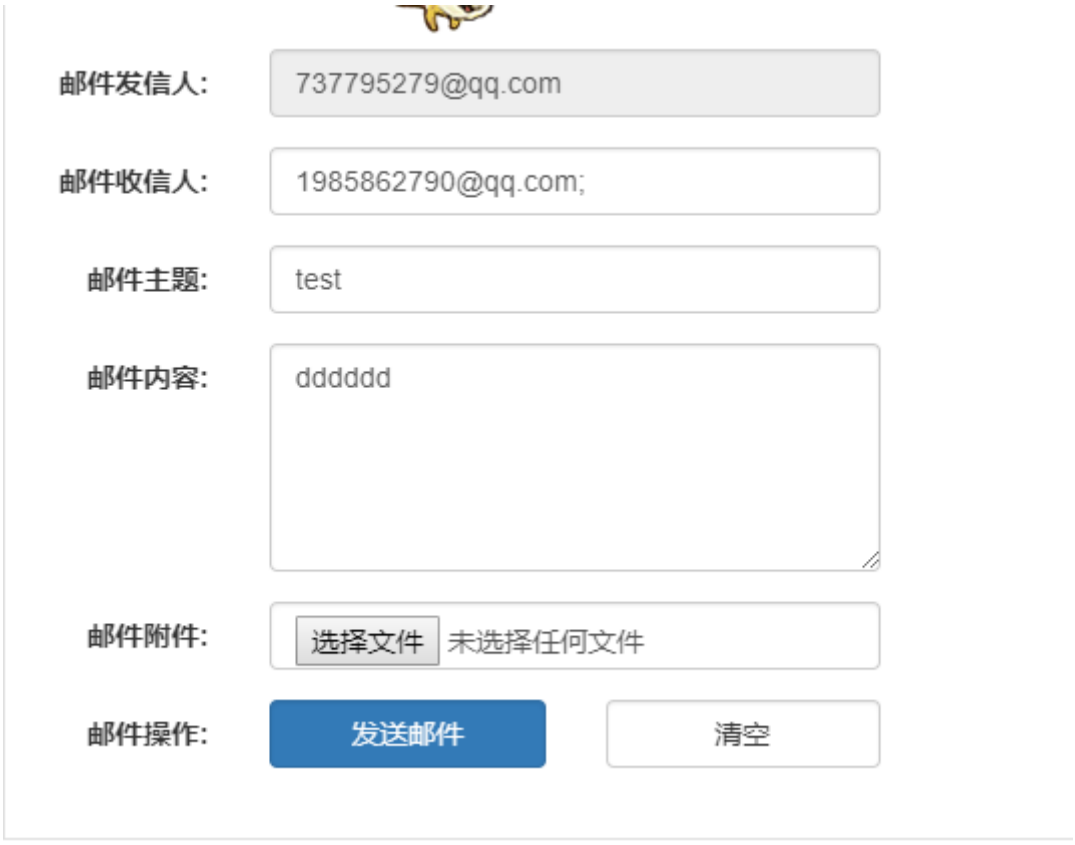
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <meta charset="UTF-8"/>
    <title>发送邮件</title>
    <link th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" rel="styl
esheet" type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"></script>
    <script th:href="@{/webjars/bootstrap/js/bootstrap.min.js}"></script>

</head>

<body>
<div class="col-md-6" style="margin:20px;padding:20px;border: #E0E0E0 1px
 solid;">
    <marquee behavior="alternate" onfinish="alert(12)" id="mq"
             onMouseOut="this.start();$('#egg').text('嗯 真听话! ');"
             onMouseOver="this.stop();$('#egg').text('有本事放开我呀! ');">
        <h5 id="egg">祝大家新年快乐! </h5><img id="doge" src="http://pics.s
c.chinaz.com/Files/pic/faces/3709/7.gif" alt="">
    </marquee>

    <form class="form-horizontal" id="mailForm">
        <div class="form-group">
            <label class="col-md-2 control-label">邮件发信人:</label>
            <div class="col-md-6">
                <input class="form-control" id="from" name="from" th:valu
e="${from}" readonly="readonly">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-2 control-label">邮件收信人:</label>
            <div class="col-md-6">
                <input class="form-control" id="to" name="to" title="多个
邮箱使用,隔开">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-2 control-label">邮件主题:</label>
            <div class="col-md-6">
```

```
                <input class="form-control" id="subject" name="subject">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-2 control-label">邮件内容:</label>
            <div class="col-md-6">
                <textarea class="form-control" id="text" name="text"
rows="5"></textarea>
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-2 control-label">邮件附件:</label>
            <div class="col-md-6">
                <input class="form-control" id="files" name="files"
type="file" multiple="multiple">
            </div>
        </div>
        <div class="form-group">
            <label class="col-md-2 control-label">邮件操作:</label>
            <div class="col-md-3">
                <a class="form-control btn btn-primary" onclick="sendMail
()">发送邮件</a>
            </div>
            <div class="col-md-3">
                <a class="form-control btn btn-default" onclick="clearFor
m()">清空</a>
            </div>
        </div>
    </form>

    <script th:inline="javascript">
        var appCtx = [[${#request.getContextPath()}]];

        function sendMail() {

            var formData = new FormData($('#mailForm')[0]);
            $.ajax({
                url: appCtx + '/mail/send',
                type: "POST",
                data: formData,
                contentType: false,
                processData: false,
                success: function (result) {
                    alert(result.status === 'ok' ? "发送成功！" : "你被Doge
嘲讽了：" + result.error);
                },
                error: function () {
                    alert("发送失败！");
                }
            });
        }
```

```
        function clearForm() {
            $('#mailForm')[0].reset();
        }

        setInterval(function () {
            var total = $('#mq').width();
            var width = $('#doge').width();
            var left = $('#doge').offset().left;
            if (left <= width / 2 + 20) {
                $('#doge').css('transform', 'rotateY(180deg)')
            }
            if (left >= total - width / 2 - 40) {
                $('#doge').css('transform', 'rotateY(-360deg)')
            }
        });
    </script>
</div>
</body>
</html>
```

四、测试发送邮件

启动工程并访问：http://localhost:8080



五、常见失败编码

553

- 553 Requested action not taken: NULL sender is not allowed 不允许发件人为空，请使用真实发件人发送；
- 553 Requested action not taken: Local user only SMTP类型的机器只允许发信人是本站用户；
- 553 Requested action not taken: no smtp MX only MX类型的机器不允许发信人是本站用户；
- 553 authentication is required SMTP需要身份验证，请检查客户端设置；

552

- 552 Illegal Attachment 不允许发送该类型的附件，包括以.uu .pif .scr .mim .hqx .bhx .cmd .vbs .bat .com .vbe .vb .js .wsh等结尾的附件；
- 552 Requested mail action aborted: exceeded mailsize limit 发送的信件大小超过了网易邮箱允许接收的最大限制；

554

- 554 DT:SPM 发送的邮件内容包含了未被许可的信息，或被系统识别为垃圾邮件。请检查是否有用户发送病毒或者垃圾邮件；
- 554 DT:SUM 信封发件人和信头发件人不匹配；
  554 IP is rejected, smtp auth error limit exceed 该IP验证失败次数过多，被临时禁止连接。请检查验证信息设置；

- 554 HL:IHU 发信IP因发送垃圾邮件或存在异常的连接行为，被暂时挂起。请检测发信IP在历史上的发信情况和发信程序是否存在异常；
- 554 HL:IPB 该IP不在网易允许的发送地址列表里；
- 554 MI:STC 发件人当天内累计邮件数量超过限制，当天不再接受该发件人的投信。请降低发信频率；
- 554 MI:SPB 此用户不在网易允许的发信用户列表里；
- 554 IP in blacklist 该IP不在网易允许的发送地址列表里。