

# 邮件发送客户端

## 1 协议简介

- 什么是SMTP?  
SMTP全称为Simple Mail Transfer Protocol（简单邮件传输协议），它是一组用于从源地址到目的地地址传输邮件的规范，通过它来控制邮件的中转方式。SMTP认证要求必须提供账号和密码才能登陆服务器，其设计目的在于避免用户受到垃圾邮件的侵扰。
- 什么是IMAP?  
IMAP全称为Internet Message Access Protocol（互联网邮件访问协议），IMAP允许从邮件服务器上获取邮件的信息、下载邮件等。IMAP与POP类似，都是一种邮件获取协议。
- 什么是POP3?  
POP3全称为Post Office Protocol 3（邮局协议），POP3支持客户端远程管理服务器端的邮件。POP3常用于“离线”邮件处理，即允许客户端下载服务器邮件，然后服务器上的邮件将会被删除。目前很多POP3的邮件服务器只提供下载邮件功能，服务器本身并不删除邮件，这种属于改进版的POP3协议。
- IMAP和POP3协议有什么不同呢？  
两者最大的区别在于，IMAP允许双向通信，即在客户端的操作会反馈到服务器上，例如在客户端收取邮件、标记已读等操作，服务器会跟着同步这些操作。而对于POP3协议虽然也允许客户端下载服务器邮件，但是在客户端的操作并不会同步到服务器上面的，例如在客户端收取或标记已读邮件，服务器不会同步这些操作。

## 2 配置邮箱

这里使用网易邮箱为例，首先登录网易邮箱，在设置中打开并勾选POP3/SMTP/IMAP服务，然后会得到一个授权码，这个邮箱和授权码将用作登陆认证。

打开设置页面



设置页面开启设置



成功获得授权码效果截图



下面是成功开启后的设置截图

**POP3/SMTP/IMAP**

开启服务：

IMAP/SMTP服务

已开启 | [关闭](#)

POP3/SMTP服务

已开启 | [关闭](#)

POP3/SMTP/IMAP服务能让你在本地客户端上收发邮件，[了解更多 >](#)

温馨提示：在第三方登录网易邮箱，可能存在邮件泄露风险，甚至危害Apple或其他平台账户安全

收取选项：

☒ 收取最近30天邮件

☐ 收取全部邮件

温馨提示：收取大量邮件，会耗费您更多的流量，建议您选择“收取最近30天邮件”

通知提醒：

☒ 开启客户端删除邮件提醒

当邮件客户端大量删除邮件时，系统会发送提醒信息

授权密码管理：

授权码是用于登录第三方邮件客户端的专用密码。

适用于登录以下服务：您开启的服务（例如POP3/IMAP/SMTP）、Exchange/CardDAV/CalDAV服务。

使用设备	启用时间	操作
		<a href="#">删除</a>

新增授权密码

## 3 Java中使用

### 3.1 Spring提供的API

- 什么是JavaMailSender和JavaMailSenderImpl？  
JavaMailSender和JavaMailSenderImpl 是Spring官方提供的集成邮件服务的接口和实现类，以简单高效的设计著称，目前是Java后端发送邮件和集成邮件服务的主流工具。

- 如何通过JavaMailSenderImpl发送邮件？  
直接在业务类注入JavaMailSenderImpl并调用send方法发送邮件。其中简单邮件可以通过SimpleMailMessage来发送邮件，而复杂的邮件（例如添加附件）可以借助MimeMessageHelper来构建MimeMessage发送邮件。例如

```
1  @Autowired
2  private JavaMailSenderImpl mailSender;
3
4  public void sendMail() throws MessagingException {
5      //简单邮件
6      SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
7      simpleMailMessage.setFrom("发送邮箱");
8      simpleMailMessage.setTo("收件邮箱");
9      simpleMailMessage.setSubject("主题");
10     simpleMailMessage.setText("内容");
11     mailSender.send(simpleMailMessage);
12
13     //复杂邮件
14     MimeMessage mimeMessage = mailSender.createMimeMessage();
15     MimeMessageHelper messageHelper = new MimeMessageHelper(mimeMessage);
16     messageHelper.setFrom("发送邮箱");
17     messageHelper.setTo("收件邮箱");
18     messageHelper.setSubject("主题");
19     messageHelper.setText("内容");
20     //附件
21     messageHelper.addInline("doge.gif", new File("xx/xx/doge.gif"));
22     messageHelper.addAttachment("work.docx", new File("xx/xx/work.docx"));
23     mailSender.send(mimeMessage);
24 }
```

## 3.2 项目中导入依赖

```
1  <!--mail-->
2  <dependency>
3      <groupId>org.springframework.boot</groupId>
4      <artifactId>spring-boot-starter-mail</artifactId>
5  </dependency>
```

## 3.3 修改配置文件

配置中加入

```
1  # 邮件配置
2  # 服务器地址（这里使用网易邮箱）
3  spring.mail.host=smtp.163.com
4  #邮件发送人账号
5  spring.mail.username=XXXXXX@163.com
6  #邮件服务授权码
7  spring.mail.password=授权码
8  #邮件发信人（即真实邮箱，需要和授权邮箱一致）
9  spring.mail.properties.from=XXXXX@163.com
10 spring.mail.properties.name=awei
```

## 3.4 构建邮件数据对象类

```
1  /**
2   * <p>
3   * 描述：书写一个邮件发送消息数据模型
4   * </p>
5   * <p>版权：&copy;01星球</p>
6   * <p>地址：01星球总部</p>
7   * @author 阿伟学长
8   * @version 1.0.0
9   */
10 @Data
11 public class MailMessage {
12     /**
13      * 多个联系人分割字符
14      */
15     protected static final String SR = ",";
16     /**
17      * 邮件发送人邮箱
18      */
19     private String from;
20     /**
21      * 邮件发送人名称
22      */
23     private String fromName;
24     /**
25      * 邮件接收人（多个邮箱则用逗号","隔开），格式如：xxx@qq.com,李明<xxxx@163.com>
26      */
27     private String to;
28     /**
29      * 抄送（多个邮箱则用逗号","隔开），格式如：xxx@qq.com,李明<xxxx@163.com>
30      */
31     private String cc;
32     /**
33      * 密送（多个邮箱则用逗号","隔开），格式如：xxx@qq.com,李明<xxxx@163.com>
34      */
35     private String bcc;
36     /**
37      * 邮件主题
38      */
39     private String subject;
40     /**
41      * 邮件内容
42      */
43     private String text;
44     /**
45      * 邮件附件
46      */
47     @JsonIgnore
48     private MultipartFile[] multipartFiles;
49     /**
50      * 发送时间
51      */
52     private Date sentDate;
53     /**
54      * 状态
```

```

55     */
56     private String status;
57     /**
58     * 报错信息
59     */
60     private String error;
61 }

```

### 3.5 书写邮件发送组件

```

1  /**
2  * <p>
3  * 描述：书写一个邮件发送组件
4  * </p>
5  * <p>版权： &copy;01星球</p>
6  * <p>地址： 01星球总部</p>
7  * @author 阿伟学长
8  * @version 1.0.0
9  */
10 @Component
11 public class MailComponent {
12     /**
13     * 注入邮件工具类
14     */
15     @Resource
16     private JavaMailSenderImpl mailSender;
17     /**
18     * 从配置中获取邮件发送人邮箱
19     * @return 邮件发送人邮箱
20     */
21     private String getMailSendFrom() {
22         return mailSender.getJavaMailProperties().getProperty("from");
23     }
24     /**
25     * 从配置中获取邮件发送人名称
26     * @return 邮件发送人名称
27     */
28     private String getMailSendFromName() {
29         return mailSender.getJavaMailProperties().getProperty("name");
30     }
31     /**
32     * 检测邮件信息类
33     * @param msg 信息对象
34     */
35     private void checkMail(MailMessage msg) {
36         if (StringUtils.isEmpty(msg.getTo())) {
37             throw new RuntimeException("邮件收信人不能为空");
38         }
39         if (StringUtils.isEmpty(msg.getSubject())) {
40             throw new RuntimeException("邮件主题不能为空");
41         }
42         if (StringUtils.isEmpty(msg.getText())) {
43             throw new RuntimeException("邮件内容不能为空");
44         }
45     }
46     /**

```

```

47      * 构建复杂邮件信息类
48      * @param msg 信息对象
49      */
50      private void sendMimeMail(MailMessage msg) {
51          try {
52              //true表示支持复杂类型
53              MimeMessageHelper messageHelper = new
MimeMessageHelper(mailSender.createMimeMessage(), true);
54              //邮件发信人从配置项读取
55              msg.setFrom(getMailSendFrom());
56              msg.setFromName(getMailSendFromName());
57              //邮件发信人
58              messageHelper.setFrom(msg.getFrom(), msg.getFromName());
59              //邮件收信人
60              for (String one : msg.getTo().split(MailMessage.SR)) {
61                  messageHelper.addTo(one);
62              }
63              //邮件主题
64              messageHelper.setSubject(msg.getSubject());
65              //邮件内容
66              messageHelper.setText(msg.getText());
67              //抄送
68              if (!StringUtils.isEmpty(msg.getCc())) {
69                  for (String one : msg.getCc().split(MailMessage.SR)) {
70                      messageHelper.addCc(one);
71                  }
72              }
73              //密送
74              if (!StringUtils.isEmpty(msg.getBcc())) {
75                  for (String one : msg.getBcc().split(MailMessage.SR)) {
76                      messageHelper.addCc(one);
77                  }
78              }
79              //添加邮件附件
80              if (msg.getMultipartFiles() != null) {
81                  for (MultipartFile multipartFile : msg.getMultipartFiles()) {
82                      messageHelper.addAttachment(Objects.requireNonNull(multipartFile.getOriginalFilename()), multipartFile);
83                  }
84              }
85              //发送时间
86              if (StringUtils.isEmpty(msg.getSentDate())) {
87                  msg.setSentDate(new Date());
88              }
89              messageHelper.setSentDate(msg.getSentDate());
90              //正式发送邮件
91              mailSender.send(messageHelper.getMimeMessage());
92              msg.setStatus("ok");
93          } catch (Exception e) {
94              //发送失败
95              throw new RuntimeException(e);
96          }
97      }
98      /**
99      * 发送邮件
100      * @param msg 邮件信息对象
101      * @return MailMessage对象，其中包含了状态信息

```

```

102     */
103     public MailMessage sendMail(MailMessage msg) {
104         try {
105             //1.检测邮件
106             checkMail(msg);
107             //2.发送邮件
108             sendMimeMail(msg);
109         } catch (Exception e) {
110             e.printStackTrace();
111             msg.setStatus("fail");
112             msg.setError(e.getMessage());
113         }
114         return msg;
115     }
116 }

```

## 3.6 测试邮件发送

书写一个Controller来测试邮件发送

```

1  /**
2   * <p>
3   * 描述：书写一个邮件发送controller来测试邮件发送
4   * </p>
5   * <p>版权：&copy;01星球</p>
6   * <p>地址：01星球总部</p>
7   * @author 阿伟学长
8   * @version 1.0.0
9   */
10 @RestController
11 @RequestMapping("mail")
12 @Api(tags = "测试邮件发送")
13 public class MailController {
14     @Resource
15     private MailComponent mailComponent;
16     /**
17      * 发送邮件
18      */
19     @ApiOperation(value = "测试邮件发送")
20     @PostMapping(value = "send")
21     public MailMessage sendMail(MailDTO mailDto) {
22         MailMessage msg = new MailMessage();
23         BeanUtil.copyProperties(mailDto, msg);
24         //发送邮件和附件
25         return mailComponent.sendMail(msg);
26     }
27 }

```

通过通过knife4j测试



POST

/mail/send

发送

1 请求头部

请求参数

AfterScript

x-www-form-urlencoded

form-data

raw

<input checked="" type="checkbox"/>	参数名称	类型	参数值	操作
<input checked="" type="checkbox"/>	bcc	文本	xxx@qq.com,李明<xxxx@163.com>	删除
<input checked="" type="checkbox"/>	cc	文本	xxx@qq.com,李明<xxxx@163.com>	删除
<input checked="" type="checkbox"/>	multipartFiles	文件	<div>选择文件</div>	删除
<input checked="" type="checkbox"/>	subject	文本	测试邮件主题	删除
<input checked="" type="checkbox"/>	text	文本	测试邮件内容	删除
<input checked="" type="checkbox"/>	to	文本	xxx@qq.com,李明<xxxx@163.com>	删除

按照格式填写参数即可，执行成功后示意效果如下

x-www-form-urlencoded

form-data

raw

<input checked="" type="checkbox"/>	参数名称	类型	参数值	操作
<input type="checkbox"/>	bcc	文本	xxx@qq.com,李明<xxxx@163.com>	删除
<input type="checkbox"/>	cc	文本	xxx@qq.com,李明<xxxx@163.com>	删除
<input checked="" type="checkbox"/>	multipartFiles	文件	<div>1607068742033.jpg</div> <div>选择文件</div>	删除
<input checked="" type="checkbox"/>	subject	文本	测试邮件主题	删除
<input checked="" type="checkbox"/>	text	文本	测试邮件内容	删除
<input checked="" type="checkbox"/>	to	文本	xxx@qq.com	删除

响应内容

Raw

Headers

Curl

☒ 显示说明 响应码: 200 耗时: 628ms 大小:

```
1- {
2-   "from": "xxx@qq.com",
3-   "fromname": "awe1",
4-   "to": "xxx@qq.com",
5-   "subject": "测试邮件主题",
6-   "text": "测试邮件内容",
7-   "sentDate": "2020-06-30 16:39:04",
8-   "status": "ok"
9- }
```

查看邮箱中内容，下面示意效果

< 返回

回复

回复全部

转发

删除

彻底删除

置顶

标记为

移至

拒收

举报

测试邮件主题 ☆

aw

awe1 发给 我

为了保护您的安全，此邮件的部分图片未下载。

测试邮件内容

附件 1

1607068742033.jpg

42.58K

快捷回复给: awe1

## 4.1 环境要求

在c++ 可以使用libcurl库简化实现邮件发送功能，这里我们主要演示龙蜥系统下面实现邮件发送功能，首先需要确保你的系统安装了libcurl库，可以使用下面命令安装

```
1 dnf install -y libcurl-devel
```

## 4.2 书写工具类

书写一个base64工具类，主要用于后面对文件进行编码

```
1 #pragma once
2 /*
3  Copyright Zero One Star. All rights reserved.
4
5  @Author: awei
6  @Date: 2023/09/23 16:46:06
7
8  Licensed under the Apache License, Version 2.0 (the "License");
9  you may not use this file except in compliance with the License.
10 You may obtain a copy of the License at
11
12     https://www.apache.org/licenses/LICENSE-2.0
13
14 Unless required by applicable law or agreed to in writing, software
15 distributed under the License is distributed on an "AS IS" BASIS,
16 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
17 See the License for the specific language governing permissions and
18 limitations under the License.
19 */
20 #ifndef _BASE64_H_
21 #define _BASE64_H_
22 #include <string>
23 std::string base64_encode(unsigned char const*, unsigned int len);
24 std::string base64_decode(std::string const& s);
25 #endif // !_BASE64_H_
```

```
1 /*
2  Copyright Zero One Star. All rights reserved.
3
4  @Author: awei
5  @Date: 2023/09/23 16:46:38
6
7  Licensed under the Apache License, Version 2.0 (the "License");
8  you may not use this file except in compliance with the License.
9  You may obtain a copy of the License at
10
11     https://www.apache.org/licenses/LICENSE-2.0
12
13 Unless required by applicable law or agreed to in writing, software
14 distributed under the License is distributed on an "AS IS" BASIS,
15 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
16 See the License for the specific language governing permissions and
```

```

17 limitations under the License.
18 */
19 #include "base64.h"
20 #include <iostream>
21
22 static const std::string base64_chars =
23 "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
24 "abcdefghijklmnopqrstuvwxyz"
25 "0123456789+/";
26
27 static inline bool is_base64(unsigned char c)
28 {
29     return (isalnum(c) || (c == '+') || (c == '/'));
30 }
31
32 std::string base64_encode(unsigned char const* bytes_to_encode, unsigned int
in_len)
33 {
34     std::string ret;
35     int i = 0, j = 0;
36     unsigned char char_array_3[3], char_array_4[4];
37
38     while (in_len--)
39     {
40         char_array_3[i++] = *(bytes_to_encode++);
41         if (i == 3)
42         {
43             char_array_4[0] = (char_array_3[0] & 0xfc) >> 2;
44             char_array_4[1] = ((char_array_3[0] & 0x03) << 4) +
((char_array_3[1] & 0xf0) >> 4);
45             char_array_4[2] = ((char_array_3[1] & 0x0f) << 2) +
((char_array_3[2] & 0xc0) >> 6);
46             char_array_4[3] = char_array_3[2] & 0x3f;
47
48             for (i = 0; (i < 4); i++)
49                 ret += base64_chars[char_array_4[i]];
50             i = 0;
51         }
52     }
53
54     if (i)
55     {
56         for (j = i; j < 3; j++)
57             char_array_3[j] = '\0';
58
59         char_array_4[0] = (char_array_3[0] & 0xfc) >> 2;
60         char_array_4[1] = ((char_array_3[0] & 0x03) << 4) + ((char_array_3[1] &
0xf0) >> 4);
61         char_array_4[2] = ((char_array_3[1] & 0x0f) << 2) + ((char_array_3[2] &
0xc0) >> 6);
62         char_array_4[3] = char_array_3[2] & 0x3f;
63
64         for (j = 0; (j < i + 1); j++)
65             ret += base64_chars[char_array_4[j]];
66
67         while ((i++ < 3))
68             ret += '=';
69

```

```

70     }
71     return ret;
72 }
73
74 std::string base64_decode(std::string const& encoded_string)
75 {
76     int in_len = encoded_string.size();
77     int i = 0, j = 0, in_ = 0;
78     unsigned char char_array_4[4], char_array_3[3];
79     std::string ret;
80
81     while (in_len-- && (encoded_string[in_] != '=') &&
is_base64(encoded_string[in_]))
82     {
83         char_array_4[i++] = encoded_string[in_]; in_++;
84         if (i == 4) {
85             for (i = 0; i < 4; i++)
86                 char_array_4[i] = base64_chars.find(char_array_4[i]);
87
88             char_array_3[0] = (char_array_4[0] << 2) + ((char_array_4[1] & 0x30) >> 4);
89             char_array_3[1] = ((char_array_4[1] & 0xf) << 4) + ((char_array_4[2] & 0x3c) >> 2);
90             char_array_3[2] = ((char_array_4[2] & 0x3) << 6) + char_array_4[3];
91
92             for (i = 0; (i < 3); i++)
93                 ret += char_array_3[i];
94             i = 0;
95         }
96     }
97
98     if (i)
99     {
100         for (j = i; j < 4; j++)
101             char_array_4[j] = 0;
102
103         for (j = 0; j < 4; j++)
104             char_array_4[j] = base64_chars.find(char_array_4[j]);
105
106         char_array_3[0] = (char_array_4[0] << 2) + ((char_array_4[1] & 0x30) >> 4);
107         char_array_3[1] = ((char_array_4[1] & 0xf) << 4) + ((char_array_4[2] & 0x3c) >> 2);
108         char_array_3[2] = ((char_array_4[2] & 0x3) << 6) + char_array_4[3];
109
110         for (j = 0; (j < i - 1); j++)
111             ret += char_array_3[j];
112     }
113
114     return ret;
115 }

```

## 书写邮件发送工具类

```

1 #pragma once
2 /*

```

```

3 Copyright Zero One Star. All rights reserved.
4
5 @Author: awei
6 @Date: 2023/09/23 21:57:30
7
8 Licensed under the Apache License, Version 2.0 (the "License");
9 you may not use this file except in compliance with the License.
10 You may obtain a copy of the License at
11
12     https://www.apache.org/licenses/LICENSE-2.0
13
14 Unless required by applicable law or agreed to in writing, software
15 distributed under the License is distributed on an "AS IS" BASIS,
16 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
17 See the License for the specific language governing permissions and
18 limitations under the License.
19 */
20 #ifndef _EMAIL_SENDER_H_
21 #define _EMAIL_SENDER_H_
22 #include <string>
23 #include <vector>
24 #include <utility>
25 #include <curl/curl.h>
26
27 /**
28  * 书写一个邮件发送工具类
29  */
30 class EmailSender
31 {
32 public:
33     //*****
34     // Method:    EmailSender
35     // FullName:  EmailSender::EmailSender
36     // Access:    public
37     // Returns:
38     // Qualifier: 构造初始化
39     // Parameter: const std::string& smtp_server 邮件服务器, 如smtp.163.com
40     // Parameter: const int smtp_port 服务器端口, 如25
41     // Parameter: const std::string& password 授权密码
42     // Parameter: const std::string& from_email 邮件发送人邮箱地址
43     // Parameter: const std::string& from_name 邮件发送人名称, 默认值No-Reply
44     // Parameter: const std::string& charset 内容编码, 默认值gb2312
45     //*****
46     EmailSender(
47         const std::string& smtp_server,
48         const int smtp_port,
49         const std::string& password,
50         const std::string& from_email,
51         const std::string& from_name = "No-Reply",
52         const std::string& charset = "gb2312");
53     ~EmailSender();
54     // 设置邮件主题和内容, 可以是HTML格式或纯文本
55     void setEmailContent(const std::string& subject = "", const std::string& body
56 = "");
57     // 添加邮件接收人
58     void addRecvEmailAddr(const std::string& email_addr, const std::string& name
59 = "");
60     // 添加邮件抄送人

```

```

59     void addCcEmailAddr(const std::string& email_addr, const std::string& name =
    "");
60     // 添加附件
61     void addAttachment(const std::string& filename);
62     // 执行发送
63     bool send();
64 private:
65     // smtp服务器
66     std::string m_smtp_url;
67     // 内容编码, 默认gb2312
68     std::string m_charset;
69     // 邮件发送人 key 邮件地址 val 发送人名称
70     std::pair<std::string, std::string> m_from;
71     // 邮件服务器授权密码
72     std::string m_password;
73     // 邮件接收人 key 邮件地址 val 接收人名称
74     std::vector<std::pair<std::string, std::string>> m_recvs;
75     // 邮件抄送人 key 邮件地址 val 抄送人名称
76     std::vector<std::pair<std::string, std::string>> m_ccs;
77     // 邮件主题
78     std::string m_email_subject;
79     // 邮件内容
80     std::string m_email_body;
81     // 邮件附件文件列表 (文件的绝对或相对路径)
82     std::vector<std::string> m_attachments;
83     // 回调函数, 将MIME协议的拼接的字符串由libcurl发出
84     static size_t payloadSource(void* ptr, size_t size, size_t nmemb, void*
    stream);
85     // 创建邮件MIME内容
86     std::string generateMimeMessage();
87     // 获取附件文件名
88     void getFileName(const std::string& path, std::string& filename);
89     // 获取附件文件类型
90     void getFileContentType(const std::string& path, std::string& contentType);
91 };
92 #endif

```

```

1  /*
2  Copyright Zero One Star. All rights reserved.
3
4  @Author: awei
5  @Date: 2023/09/23 21:57:37
6
7  Licensed under the Apache License, Version 2.0 (the "License");
8  you may not use this file except in compliance with the License.
9  You may obtain a copy of the License at
10
11     https://www.apache.org/licenses/LICENSE-2.0
12
13  Unless required by applicable law or agreed to in writing, software
14  distributed under the License is distributed on an "AS IS" BASIS,
15  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
16  See the License for the specific language governing permissions and
17  limitations under the License.
18  */
19 #include "email_sender.h"

```

```

20 #include "base64.h"
21 #include <iostream>
22 #include <sstream>
23 #include <fstream>
24 #include <string.h>
25
26 EmailSender::EmailSender(const std::string& smtp_server,
27     const int smtp_port,
28     const std::string& password,
29     const std::string& from_email,
30     const std::string& from_name /*= "No-Reply"*/,
31     const std::string& charset/* = "gb2312"*/)
32 {
33     // 初始化设置
34     m_smtp_url = "smtp://" + smtp_server + ':' + std::to_string(smtp_port);
35     m_from = std::make_pair(from_email, from_name);
36     m_password = password;
37     m_charset = charset;
38 }
39
40 EmailSender::~EmailSender()
41 {
42     m_recvs.clear();
43     m_ccs.clear();
44     m_attachments.clear();
45 }
46
47 void EmailSender::setEmailContent(const std::string& subject, const std::string&
body)
48 {
49     m_email_subject = subject;
50     m_email_body = body;
51 }
52
53 void EmailSender::addRecvEmailAddr(const std::string& email_addr, const
std::string& name)
54 {
55     m_recvs.push_back(std::make_pair(email_addr, name));
56 }
57
58 void EmailSender::addCcEmailAddr(const std::string& email_addr, const
std::string& name)
59 {
60     m_ccs.push_back(std::make_pair(email_addr, name));
61 }
62
63 void EmailSender::addAttachment(const std::string& filename)
64 {
65     m_attachments.push_back(filename);
66 }
67
68 bool EmailSender::send()
69 {
70     CURL* curl;
71     CURLcode res = CURLE_OK;
72     curl = curl_easy_init();
73     bool ret = false;
74     if (curl)

```

```

75     {
76         /* This is the URL for your mailserver */
77         curl_easy_setopt(curl, CURLOPT_URL, m_smtp_url.c_str());
78
79         /* Login smtp server to verify */
80         curl_easy_setopt(curl, CURLOPT_USERNAME, m_from.first.c_str());
81         curl_easy_setopt(curl, CURLOPT_PASSWORD, m_password.c_str());
82
83         /* If you want to connect to a site who isn't using a certificate that
is
84         * signed by one of the certs in the CA bundle you have, you can skip
the
85         * verification of the server's certificate. This makes the connection
86         * A LOT LESS SECURE.
87         *
88         * If you have a CA cert for the server stored someplace else than in
the
89         * default bundle, then the CURLOPT_CAPATH option might come handy for
you.
90         */
91 #ifdef SKIP_PEER_VERIFICATION
92     curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0L);
93 #endif
94
95     /* If the site you're connecting to uses a different host name that what
96     * they have mentioned in their server certificate's commonName (or
97     * subjectAltName) fields, libcurl will refuse to connect. You can skip
98     * this check, but this will make the connection less secure.
99     */
100 #ifdef SKIP_HOSTNAME_VERIFICATION
101     curl_easy_setopt(curl, CURLOPT_SSL_VERIFYHOST, 0L);
102 #endif
103     /* Note that this option isn't strictly required, omitting it will
result
104     * in libcurl sending the MAIL FROM command with empty sender data. All
105     * autoresponses should have an empty reverse-path, and should be
directed
106     * to the address in the reverse-path which triggered them. Otherwise,
107     * they could cause an endless loop. See RFC 5321 Section 4.5.5 for more
details.
108     */
109     std::string from_email_addr = '<' + m_from.first + '>';
110     curl_easy_setopt(curl, CURLOPT_MAIL_FROM, from_email_addr.c_str());
111
112     /* Add two recipients, in this particular case they correspond to the
113     * To: and Cc: addressees in the header, but they could be any kind of
recipient.
114     */
115     struct curl_slist* recipients = NULL;
116     for (auto& email_pair : m_recvs)
117     {
118         std::string email_addr = '<' + email_pair.first + '>';
119         recipients = curl_slist_append(recipients, email_addr.c_str());
120     }
121     for (auto& email_pair : m_ccs)
122     {
123         std::string email_addr = '<' + email_pair.first + '>';
124         recipients = curl_slist_append(recipients, email_addr.c_str());

```



```

125     }
126     curl_easy_setopt(curl, CURLOPT_MAIL_RCPT, recipients);
127
128     // 准备消息内容
129     std::stringstream stream;
130     stream.str(generateMimeMessage().c_str());
131     stream.flush();
132
133     /* We're using a callback function to specify the payload (the headers
and
134     * body of the message). You could just use the CURLOPT_READDATA option
to
135     * specify a FILE pointer to read from.
136     */
137     curl_easy_setopt(curl, CURLOPT_READFUNCTION,
&EmailSender::payloadSource);
138     curl_easy_setopt(curl, CURLOPT_READDATA, (void*)&stream);
139     curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);
140
141     /* Since the traffic will be encrypted, it is very useful to turn on
debug
142     * information within libcurl to see what is happening during the
transfer
143     */
144 #ifdef DEBUG
145     curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);
146 #else
147     curl_easy_setopt(curl, CURLOPT_VERBOSE, 0L);
148 #endif
149
150     /* Send the message */
151     res = curl_easy_perform(curl);
152     /* Check for errors */
153     if (res != CURLE_OK)
154         fprintf(stderr, "curl_easy_perform() failed: %s\n",
curl_easy_strerror(res));
155     else
156         ret = true;
157
158     /* Free the list of recipients */
159     curl_slist_free_all(recipients);
160     /* Always cleanup */
161     curl_easy_cleanup(curl);
162 }
163 return ret;
164 }
165
166 size_t EmailSender::payloadSource(void* ptr, size_t size, size_t nmemb, void*
stream)
167 {
168     size_t num_bytes = size * nmemb;
169     char* data = (char*)ptr;
170     std::stringstream* strstream = (std::stringstream*)stream;
171     strstream->read(data, num_bytes);
172     return strstream->gcount();
173 }
174
175 std::string EmailSender::generateMimeMessage()

```

```

176 {
177     std::string message;
178     // 发送人
179     message += "From: ";
180     message += m_from.second + '<' + m_from.first + '>' + "\r\n";
181     // 接收人
182     message += "To: ";
183     for (int i = 0; i < m_recvs.size(); i++)
184     {
185         message += m_recvs[i].second + '<' + m_recvs[i].first + '>';
186         if (i != m_recvs.size() - 1)
187             message += ',';
188     }
189     message += "\r\n";
190     // 抄送人
191     if (!m_ccs.empty())
192     {
193         message += "Cc: ";
194         for (int i = 0; i < m_ccs.size(); i++)
195         {
196             message += m_ccs[i].second + '<' + m_ccs[i].first + '>';
197             if (i != m_ccs.size() - 1)
198                 message += ',';
199         }
200         message += "\r\n";
201     }
202     // 主题
203     message += "Subject: ";
204     message += m_email_subject;
205     message += "\r\nMime-Version: 1.0";
206     message += "\r\nContent-Type: multipart/mixed;boundary=\"simple boundary\"";
207     message += "\r\n";
208     // 内容
209     message += "\r\n--simple boundary";
210     message += "\r\nContent-Type: text/html;charset=" + m_charset;
211     message += "\r\nContent-Transfer-Encoding: 7bit";
212     message += "\r\n\r\n"; // 注意：内容和描述信息之间必须要有一个空行不然会在网易
邮箱出现无内容bug
213     message += m_email_body;
214     message += "\r\n\r\n";
215     // 附件
216     if (!m_attachments.empty())
217     {
218         std::string filename = "";
219         std::string filetype = "";
220         for (std::string& path : m_attachments)
221         {
222             getFileName(path, filename);
223             getFileContentType(path, filetype);
224             message += "\r\n--simple boundary";
225             message += "\r\nContent-Type: " + filetype + "\tname=" + filename;
226             message += "\r\nContent-Disposition: attachment;filename=" +
filename;
227             message += "\r\nContent-Transfer-Encoding: base64";
228             message += "\r\n\r\n";
229
230             FILE* pt = NULL;
231             if ((pt = fopen(path.c_str(), "rb")) == NULL)

```

```

232         {
233             std::cerr << "open file fail: " << path << std::endl;
234             continue;
235         }
236         fseek(pt, 0, SEEK_END);
237         int len = ftell(pt);
238         fseek(pt, 0, SEEK_SET);
239         int rlen = 0;
240         char buf[55];
241         for (size_t i = 0; i < len / 54 + 1; i++)
242         {
243             memset(buf, 0, 55);
244             rlen = fread(buf, sizeof(char), 54, pt);
245             message += base64_encode((const unsigned char*)buf, rlen);
246             message += "\r\n";
247         }
248         fclose(pt);
249         pt = NULL;
250     }
251     message += "\r\n--simple boundary--\r\n";
252 }
253 return message;
254 }
255
256 void EmailSender::getFileName(const std::string& path, std::string& filename)
257 {
258     auto p = path.find_last_of('/');
259     if (p == std::string::npos) p = path.find_last_of('\\');
260     if (p != std::string::npos)
261     {
262         p += 1;
263         filename = path.substr(p, path.length() - p);
264     }
265     std::string tmp = "?";
266     tmp += m_charset;
267     tmp += "?B?";
268     tmp += base64_encode((unsigned char*)filename.c_str(), filename.size());
269     tmp += "?=";
270     filename = "\"" + tmp + "\"";
271 }
272
273 void EmailSender::getFileContentType(const std::string& path, std::string&
contentType)
274 {
275     // 获取文件后缀
276     std::string suffix = "";
277     auto p = path.find_last_of('.');
278     if (p != std::string::npos)
279     {
280         p += 1;
281         suffix = path.substr(p, path.length() - p);
282     }
283     // 根据后缀设置contentType
284     if (suffix == "txt") contentType = "plain/text";
285     else if (suffix == "xml") contentType = "text/xml";
286     else if (suffix == "html") contentType = "text/html";
287     else if (suffix == "jpeg" || suffix == "jpg") contentType = "image/jpeg";
288     else if (suffix == "png") contentType = "image/png";

```

```

289     else if (suffix == "gif") contentType = "image/gif;";
290     else if (suffix == "exe") contentType = "application/x-msdownload;";
291     else contentType = "application/octet-stream;";
292 }

```

## 4.3 发送调用示例

```

1  #include <iostream>
2  #include "email_sender.h"
3  int main()
4  {
5      std::string email_subject = u8"测试主题";
6      std::string email_body1 = "<html><meta charset=\"UTF-8\"><body>"
7          u8"<p>我是<b>中文</b></p>"
8          u8"<p><font color=red>我是红色的中文</font></p>"
9          "</body></html>";
10     std::string email_body2 = u8"我是中文，文字消息内容。";
11     EmailSender email_sender("smtp.163.com", 25, "授权码", "授权邮箱");
12     email_sender.addRecvEmailAddr("xxxxx@qq.com", u8"大明");
13     email_sender.addRecvEmailAddr("xxxx@163.com", u8"二明");
14     email_sender.addCcEmailAddr("xxxxxxx@qq.com", u8"小舞");
15     email_sender.setEmailContent(email_subject, email_body1);
16     //email_sender.addAttachment("/home/aliyun/1.jpg");
17     if (email_sender.send())
18         std::cout << "mail send ok" << std::endl;
19     else
20         std::cout << "mail send fail" << std::endl;
21     return 0;
22 }

```

## 4.4 cmake项目配置

需要再你的cmake配置中引入libcurl库

```

1  # 检查CURL库是否存在
2  find_package (CURL REQUIRED)
3  if (CURL_FOUND)
4      include_directories (${CURL_INCLUDE_DIRS})
5      message (STATUS "CURL Found!")
6  endif()
7
8  # 将源代码添加到此项目的可执行文件。
9  add_executable (${appName} ..... "email_sender.cpp" "base64.cpp")
10
11 # 链接动态库
12 target_link_libraries (${appName} CURL::libcurl)

```

## 4.5 测试发送

执行示例发送代码，然后观察你的邮箱中是否收到邮件