rubytomato@github
@rubytomato@github
2018 年 04 月 24 日に更新
Java アプリケーションから E メールを送信するサンプルコード
https://qiita.com/rubytomato@github/items/b106ff8011bcad60bce2

Java
spring-boot
commons-email
この記事は最終更新日から 1 年以上が経過しています。
概要
E メールを送信する Java アプリケーションのサンプルコードです。JavaMail、Commons Email、Spring Boot Starter mail の 3 つのライブラリを試しました。

環境

Windows7 (64bit)
Java 1.8.0_65
JavaMail 1.5.4
Spring Boot 1.3.0
spring-boot-starter-mail 1.3.0
Commons Email 1.4
参考

Testing mail code in Spring Boot application
Sending email
E-mail 送信(SMTP)
JavaMail
準備
pom.xml

```
<dependency>
  <groupId>com.sun.mail</groupId>
  <artifactId>javax.mail</artifactId>
  <version>1.5.4</version>
</dependency>
```

properties
Package javax.mail

```
mail.user=
mail.host=
mail.from=
mail.mime.address.strict=true
```

mail.store.protocol=imap
mail.transport.protocol=smtp
mail.debug=false　　# debug 出力
Package com.sun.mail.smtp

An SMTP protocol provider for the JavaMail API that provides access to an SMTP server.

mail.smtp.user=
mail.smtp.host=
mail.smtp.port=
mail.smtp.from=
mail.smtp.connectiontimeout=0　　# コネクション確立までのタイムアウト時間(ミリ秒)
mail.smtp.timeout=0　　# SMTP サーバとの通信(read)のタイムアウト時間(ミリ秒)
mail.smtp.writetimeout=0　　# ソケットの書き込みのタイムアウト時間(ミリ秒)
mail.smtp.auth=false　　# auth コマンドでのユーザー認証を行うか
mail.smtp.ssl.enable=false　　#
mail.smtp.starttls.enable=false　#
mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory　　# ssl
mail.smtp.socketFactory.fallback=false　　# ssl
mail.smtp.socketFactory.port=　　# ssl
JavaMail を使う時はちゃんとタイムアウト値を設定しよう
JavaMail mail.smtp.ssl.enable is not working
サンプルコード
```
import java.io.UnsupportedEncodingException;
import java.util.Properties;

import javax.mail.Address;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

@Component
public class JavaMailSample {
   private static final Logger log = LoggerFactory.getLogger(JavaMailSample.class);
```

```
public static void main(String[] args) {
    JavaMailSample mailSend = new JavaMailSample();
    mailSend.send("JavaMail テストメール", "テストメールの本文");
}

public void send(String subject, String content) {

    final String to = "xxx.yyy.zzz@example.com";
    final String from = "*****.*****.*****@gmail.com";

    // Google account mail address
    final String username = "*****.*****.*****@gmail.com";
    // Google App password
    final String password = "***************";

    //final String charset = "ISO-2022-JP";
    final String charset = "UTF-8";

    final String encoding = "base64";

    // for gmail
    String host = "smtp.gmail.com";
    String port = "587";
    String starttls = "true";

    // for local
    //String host = "localhost";
    //String port = "2525";
    //String starttls = "false";

    Properties props = new Properties();
    props.put("mail.smtp.host", host);
    props.put("mail.smtp.port", port);
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.starttls.enable", starttls);

    props.put("mail.smtp.connectiontimeout", "10000");
    props.put("mail.smtp.timeout", "10000");

    props.put("mail.debug", "true");
```

```
        Session session = Session.getInstance(props,
        new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        });

        try {
          MimeMessage message = new MimeMessage(session);

          // Set From:
          message.setFrom(new InternetAddress(from, "Watanabe Shin"));
          // Set ReplyTo:
          message.setReplyTo(new Address[]{new InternetAddress(from)});
          // Set To:
          message.setRecipient(Message.RecipientType.TO, new InternetAddress(to));

          message.setSubject(subject, charset);
          message.setText(content, charset);

          message.setHeader("Content-Transfer-Encoding", encoding);

          Transport.send(message);

        } catch (MessagingException e) {
          throw new RuntimeException(e);
        } catch (UnsupportedEncodingException e) {
          throw new RuntimeException(e);
        }

    }

}
```

Commons Email

Users Guide

準備

pom.xml

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-email</artifactId>
  <version>1.4</version>
</dependency>
```

サンプルコード

```
package com.example.sbmail.mail;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;
import org.springframework.stereotype.Component;

@Component
public class CommonsMailSample {

   private String to = "xxx.yyy.zzz@example.com";
   private String from = "*****.*****.*****@gmail.com";

   // Google account mail address
   private String username = "*****.*****.*****@gmail.com";
   // Google app password
   private String password = "**********";

   //private String charset = "ISO-2022-JP";
   private String charset = "UTF-8";

   private String encoding = "base64";

   // for gmail
   private String host = "smtp.gmail.com";
   private int port = 587;
   private boolean starttls = true;

   // for local
   //private String host = "localhost";
   //private int port = 2525;
   //private boolean starttls = false;

   private Map<String, String> headers = new HashMap<String, String>(){
      private static final long serialVersionUID = 1L;
      {
```

```
        put("Content-Transfer-Encoding", encoding);
      }
    };

    public static void main(String[] args) throws IOException {
      CommonsMailSample sendMail = new CommonsMailSample();
      sendMail.send("Commons Email テストメール", "テストメールの本文");
    }

    public void send(String subject, String content) {

      Email email = new SimpleEmail();

      try {
        email.setHostName(host);
        email.setSmtpPort(port);
        email.setCharset(charset);
        email.setHeaders(headers);
        email.setAuthenticator(new DefaultAuthenticator(username, password));
        email.setStartTLSEnabled(starttls);
        email.setFrom(from);
        email.addTo(to);
        email.setSubject(subject);
        email.setMsg(content);
        email.setDebug(true);

        email.send();

      } catch (EmailException e) {
        e.printStackTrace();
      }
    }

}
```

spring-boot-starter-mail

準備

pom.xml

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-mail</artifactId>
  <version>1.3.0.RELEASE</version>
</dependency>
```

```
<dependency>
  <groupId>com.sun.mail</groupId>
  <artifactId>javax.mail</artifactId>
</dependency>
```

properties

MailProperties.java

application.yml

```
spring:
# Email (MailProperties)
  mail:
    default-encoding: UTF-8
    protocol: smtp
    #host: localhost
    #port: 2525
    host: smtp.gmail.com
    port: 587
    jndi-name: mail/Session
    password: ********** # Google App password
    username: *****.*****.*****@gmail.com # Google account mail address
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            #enable: false
            enable: true
          socketFactory:
            #port: 2525
            port: 587
            class: javax.net.ssl.SSLSocketFactory
            fallback: false
        debug: true
    test-connection: false
```

MailSenderAutoConfiguration.java

サンプルコード

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
```

```
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Component;

@Component
public class SpringBootMailSample {
   private static final Logger log = LoggerFactory.getLogger(SpringBootMailSample.class);

   private final JavaMailSender javaMailSender;

   @Autowired
   SpringBootMailSample(JavaMailSender javaMailSender) {
      this.javaMailSender = javaMailSender;
   }

   public SimpleMailMessage send(String subject, String content) {

      SimpleMailMessage mailMessage = new SimpleMailMessage();

      mailMessage.setTo("xxx.yyy.zzz@example.com");
      mailMessage.setReplyTo("*****.*****.*****@gmail.com");
      mailMessage.setFrom("*****.*****.*****@gmail.com");
      mailMessage.setSubject(subject);
      mailMessage.setText(content);

      javaMailSender.send(mailMessage);

      return mailMessage;
   }

}
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Component;

@Component
public class SpringBootMailSample2 {
```

```
  private static final Logger log = LoggerFactory.getLogger(SpringBootMailSample2.class);

  private final JavaMailSender javaMailSender;

  @Autowired
  public SpringBootMailSample2(JavaMailSender javaMailSender) {
    this.javaMailSender = javaMailSender;
  }

  public void send(String subject, String content) {

    try {

      MimeMessage mail = javaMailSender.createMimeMessage();

      mail.setHeader("Content-Transfer-Encoding", "base64");

      MimeMessageHelper helper = new MimeMessageHelper(mail, false);

      helper.setTo("xxx.yyy.zzz@example.com");
      helper.setReplyTo("*****.*****.*****@gmail.com");
      helper.setFrom("*****.*****.*****@gmail.com");
      helper.setSubject(subject);
      helper.setText(content);

      javaMailSender.send(mail);

    } catch (MessagingException e) {
      e.printStackTrace();
    }

  }

}
```

開発用 SMTP サーバー

ローカル環境でメールの送信テストを行いたい場合に使用する開発用 SMTP サーバーを幾つか試用しました。

FakeSMTP

Dummy SMTP server with GUI for testing emails in applications easily.

Java で開発されているので実行環境を選びません。受信したメールを eml 形式のファイルとして保存することができます。

日本語(ISO-2020-JP,UTF-8)で書かれたメールにも対応しています。

ビルド
ソースコードからビルドして jar ファイルを生成します。(ビルド済みの jar を取得して使用することもできます)

$ git clone https://github.com/Nilhcem/FakeSMTP.git
> cd FaceSMTP
> mvn package -Dmaven.test.skip
実行
-o で指定したディレクトリに受信したメールが eml ファイルとして保存されます。

> java -jar fakeSMTP-2.1-SNAPSHOT.jar -p 2525 -a 127.0.0.1 -o d:¥mail¥message
ファイルに書き出す必要がない場合は代わりに-m オプションを付けます。

> java -jar fakeSMTP-2.1-SNAPSHOT.jar -p 2525 -a 127.0.0.1 -m
上記のオプションで実行すると GUI が起動します。

FaceSMTPServer.png

メールをクリックするとメールクライアントが立ち上がりメール本文を確認することが出来ます。

FaceSMTPServer1.png

バックグラウンドモードで実行したい場合は-b、-s オプションを付けます。

> java -jar fakeSMTP-2.1-SNAPSHOT.jar -p 2525 -a 127.0.0.1 -b -s -o d:¥mail¥message
smtp4dev
Windows 7/Vista/XP/2003/2010 compatible dummy SMTP server. Sits in the system tray and does not deliver the received messages. The received messages can be quickly viewed, saved and the source/structure inspected. Useful for testing/debugging software that generates email.

Windows のネイティブアプリケーションです。smtp4dev.exe という実行ファイルのみでインストール不要です。
最終リリースが 2011 年となっているのが気になりますが今のところ問題なく利用できました。
日本語にも対応しています。(ISO-2022-JP の場合、GUI 上では件名が文字化けしましたがメールクライアントでは正常に表示されます。)

smtp4dev0.png

View ボタンをクリックするとメールクライアントでメールを表示することができます。図はありませんが Inspect ボタンをクリックするとメールヘッダーやソースを確認することができます。

mailtest0.png

option 画面

smtp4dev1.png

MailDev
MailDev is a simple way to test your projects' emails during development with an easy to use web interface that runs on your machine.

インストールと実行には Nodejs,npm が必要です。ISO-2022-JP の場合は文字化けを起こしますが、UTF-8 は問題ありませんでした。

インストール
> npm install -g maildev
実行
> maildev -s 2525 -w 8080 --ip 127.0.0.1
ブラウザベースの GUI が付属しています。

maildev0.png

SubEtha
SubEtha SMTP is a Java library for receiving SMTP mail

上記の smtp サーバーとは用途が違い、Unit Test で使用する組み込み型 smtp サーバーです。こちらは未だ試用していませんが便利そうだったので記載だけしておきます。

ユニットテスト用のメールサーバ subethasmtp があることを知りました
SubEthaMail の wiser で仮想 SMTP を立てて java のメール送信をユニットテストする！
メモ
テンプレートエンジン
The Apache Velocity Project
Apache FreeMarker
mustache.java
Handlebars.java
mustache.java

pom.xml
```
<dependency>
  <groupId>com.github.spullara.mustache.java</groupId>
  <artifactId>compiler</artifactId>
  <version>0.9.1</version>
</dependency>
```

Handlebars.java

pom.xml

```
<dependency>
  <groupId>com.github.jknack</groupId>
  <artifactId>handlebars</artifactId>
  <version>4.0.2</version>
</dependency>
```

commons-email と handlerbars.java でメール送信

Gmail

Google アカウントの 2 段階認証

smtp サーバーに smtp.gmail.com を使用してメール送信を行う場合、指定する Google アカウントが 2 段階認証プロセスを適用していると下記のエラーメッセージを出力してメール送信に失敗します。

この場合はその Google アカウントでアプリパスワードを取得し、そのパスワードを使用することでメール送信ができるようになります。

```
javax.mail.AuthenticationFailedException: 534-5.7.9 Application-specific password required. Learn more at
534 5.7.9  https://support.google.com/accounts/answer/185833 63sm19755563pfq.92 - gsmtp
```

アプリパスワードを取得する

Google アカウント情報ページの"Google へのログイン"メニューを選択し"アプリ パスワード"をクリックします。

g01.png

アプリパスワードの用途をメニューから選ぶか、任意の名前を付けてパスワードを作成します。

g02.png

図の通り(モザイクをかけています)パスワードが作成されるので、このパスワードを google アカウントのパスワードとして使用します。

g03.png

作成したパスワードはいつでも取り消すことができます。

g04.png

メール送信に使用するポート

25

SMTP

465

SMTPS (Simple Mail Transfer Protocol Secure)

ポート 465 は SMTPS に割り当てられていたが、STARTTLS の制定により割り当ては無効となった。

587

MSA (Mail Submission Agent)

MUA-MSA 間の投稿(Submission)ではポート 587 が使われることが多い。

STARTTLS
STARTTLS（スタート・ティーエルエス）は、平文の通信プロトコルを暗号化通信に拡張する方法のひとつ。
STARTTLS は、IMAP や POP3 に対しては RFC 2595、SMTP に対しては RFC 3207、FTP に対しては RFC 4217、XMPP に対しては RFC 6120 の 5 節、LDAP に対しては RFC 4511 の 4.14 節、NNTP に対しては RFC 4642 で規定する。

STARTTLS - Wikipedia

参考になった stackoverflow
Spring Boot 1.2.5.RELEASE - Sending E-mail via Gmail SMTP
spring.mail.properties.mail.smtp.ssl.enable = true
Unable to Send Rich Text Email - Thymeleaf + Spring 4

What is the difference between ports 465 and 587?

編集リクエスト

ストック

102
rubytomato@github
渡邉 真
@rubytomato@github
今まで Java をメインにやってきましたが、JavaScript(Node.js)の習得に取り組み始めました。