Email Sending

**Dependency:** Spring web, java web sender

**User.java**

**package** com.cjc.main.model;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** User {

**private** String emailAddress;

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) {

**this**.emailAddress = emailAddress;

}

}

**RegisteratioController.java**

**package** com.cjc.main.controller;

**import** javax.mail.MessagingException;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.mail.MailException;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.cjc.main.service.Mailserver;

/\*\*\* This class contains a Mail API developed using Spring Boot

\*\* **@author** MukulJaiswal

\* \*/

@RestController

**public** **class** RegistrationController {

@Autowired

**private** Mailserver notificationService;

@Autowired

**private** com.cjc.main.model.User user;

/\*\*\* **@return**\*/

@RequestMapping("send-mail")

**public** String send() {

/\*\* Creating a User with the help of User class that we have declared and setting

\* Email address of the sender.\*/

user.setEmailAddress("shruteewairagade21@gmail.com"); //Receiver's email address

/\*

\* Here we will call sendEmail() for Sending mail to the sender.

\*/

**try** {

notificationService.sendEmail(user);

} **catch** (MailException mailException) {

System.***out***.println(mailException);

}

**return** "Congratulations! Your mail has been send to the user."+"to"+user;

}

/\*\*

\*

\* **@return**

\* **@throws** MessagingException

\*/

@RequestMapping("send-mail-attachment")

**public** String sendWithAttachment() **throws** MessagingException {

/\*

\* Creating a User with the help of User class that we have declared and setting

\* Email address of the sender.

\*/

user.setEmailAddress("shruteewairagade21@gmail.com"); //Receiver's email address

/\*

\* Here we will call sendEmailWithAttachment() for Sending mail to the sender

\* that contains a attachment.

\*/

**try** {

notificationService.sendEmailWithAttachment(user);

} **catch** (MailException mailException) {

System.***out***.println(mailException);

}

**return** "Congratulations! Your mail has been send to the user.";

}

}

**Mailserver.java**

**package** com.cjc.main.service;

**import** javax.mail.MessagingException;

**import** javax.mail.internet.MimeMessage;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.mail.MailException;

**import** org.springframework.mail.SimpleMailMessage;

**import** org.springframework.mail.javamail.JavaMailSender;

**import** org.springframework.mail.javamail.MimeMessageHelper;

**import** org.springframework.stereotype.Service;

**import** com.cjc.main.model.User;

@Service

**public** **class** Mailserver {

/\*

\* The Spring Framework provides an easy abstraction for sending email by using

\* the JavaMailSender interface, and Spring Boot provides auto-configuration for

\* it as well as a starter module.

\*/

**private** JavaMailSender javaMailSender;/\*\*

\* \* **@param** javaMailSender\*/

@Autowired

**public** Mailserver(JavaMailSender javaMailSender) {

**this**.javaMailSender = javaMailSender;

}

/\*\*

\* This function is used to send mail without attachment.

\* **@param** user

\* **@throws** MailException

\*/

**public** **void** sendEmail(User user) **throws** MailException {

/\*

\* This JavaMailSender Interface is used to send Mail in Spring Boot. This

\* JavaMailSender extends the MailSender Interface which contains send()

\* function. SimpleMailMessage Object is required because send() function uses

\* object of SimpleMailMessage as a Parameter

\*/

SimpleMailMessage mail = **new** SimpleMailMessage();

mail.setTo(user.getEmailAddress());

mail.setSubject("Testing Mail API");

mail.setText("Hurray ! You have done that dude...");

/\*

\* This send() contains an Object of SimpleMailMessage as an Parameter

\*/

javaMailSender.send(mail);

}

/\*\*

\* This fucntion is used to send mail that contains a attachment.

\*

\* **@param** user

\* **@throws** MailException

\* **@throws** MessagingException

\*/

**public** **void** sendEmailWithAttachment(User user) **throws** MailException, MessagingException {

MimeMessage mimeMessage = javaMailSender.createMimeMessage();

MimeMessageHelper helper = **new** MimeMessageHelper(mimeMessage, **true**);

helper.setTo(user.getEmailAddress());

helper.setSubject("Testing Mail API with Attachment");

helper.setText("Please find the attached document below.");

ClassPathResource classPathResource = **new** ClassPathResource("2. Hibernate.pdf");

helper.addAttachment(classPathResource.getFilename(), classPathResource);

javaMailSender.send(mimeMessage);

}

}

**Application.properties**

#Make sure less secure App is enabled at link https://myaccount.google.com/lesssecureapps

server.port=8081

spring.mail.host = smtp.gmail.com

spring.mail.username = shruteewairagade21@gmail.com

spring.mail.password = \*\*\*\*\*\*\*\*

#Below Properties are needed for maintaining the session.

#Two Step verification should not be enabled for gmail.

spring.mail.properties.mail.smtp.auth = true

spring.mail.properties.mail.smtp.socketFactory.port = 465

spring.mail.properties.mail.smtp.socketFactory.class = javax.net.ssl.SSLSocketFactory

spring.mail.properties.mail.smtp.socketFactory.fallback = false

Note: Please change your Gmail setting,

Go to Manage your Google Account -> Security -> Less secure app access -> Turn on access (not recommended)