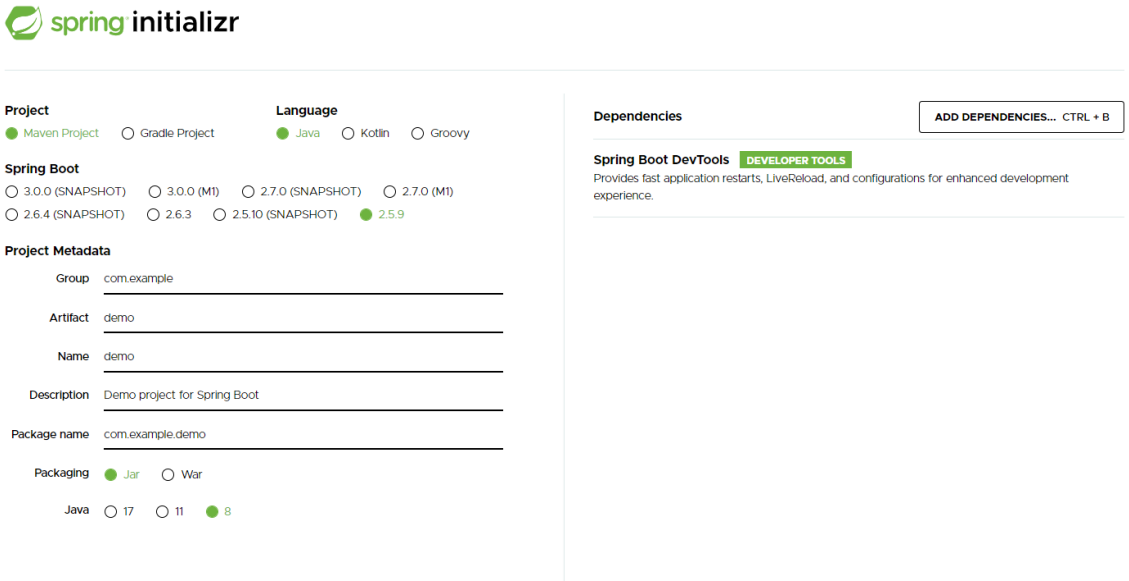
How to Send Email Using Spring Boot

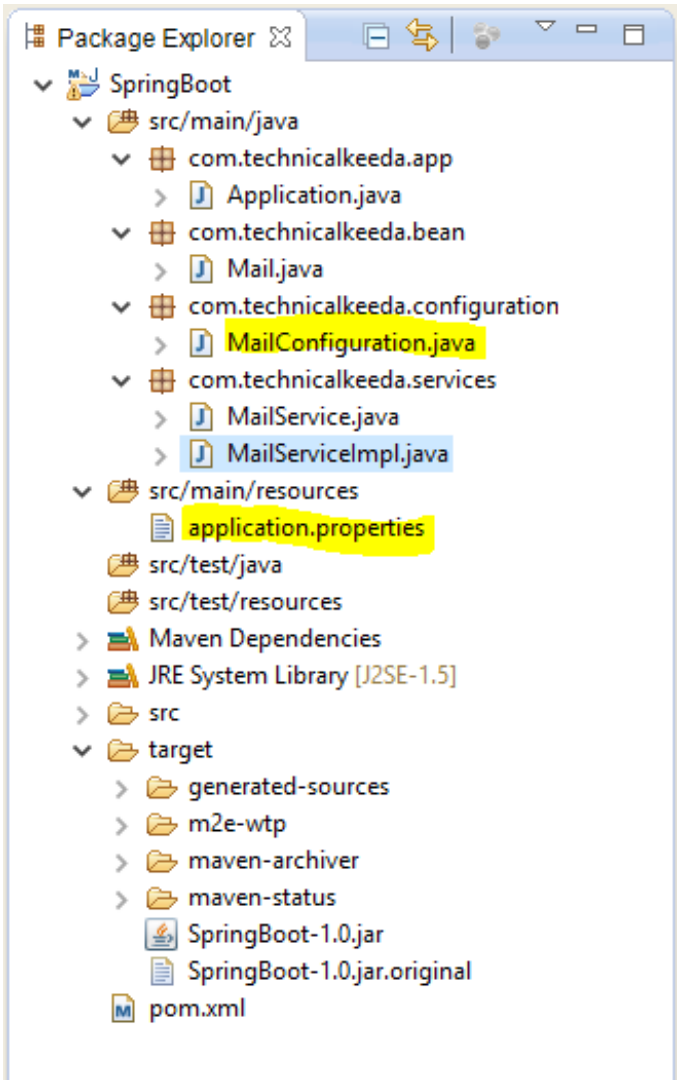
In this tutorial, You will learn how to send an Email using Spring Boot application. We will set up a maven project, will use spring-boot-starter-mail dependency, define email configurations and actual implementation of the Mail program.

#spring #springboot #springbootemail

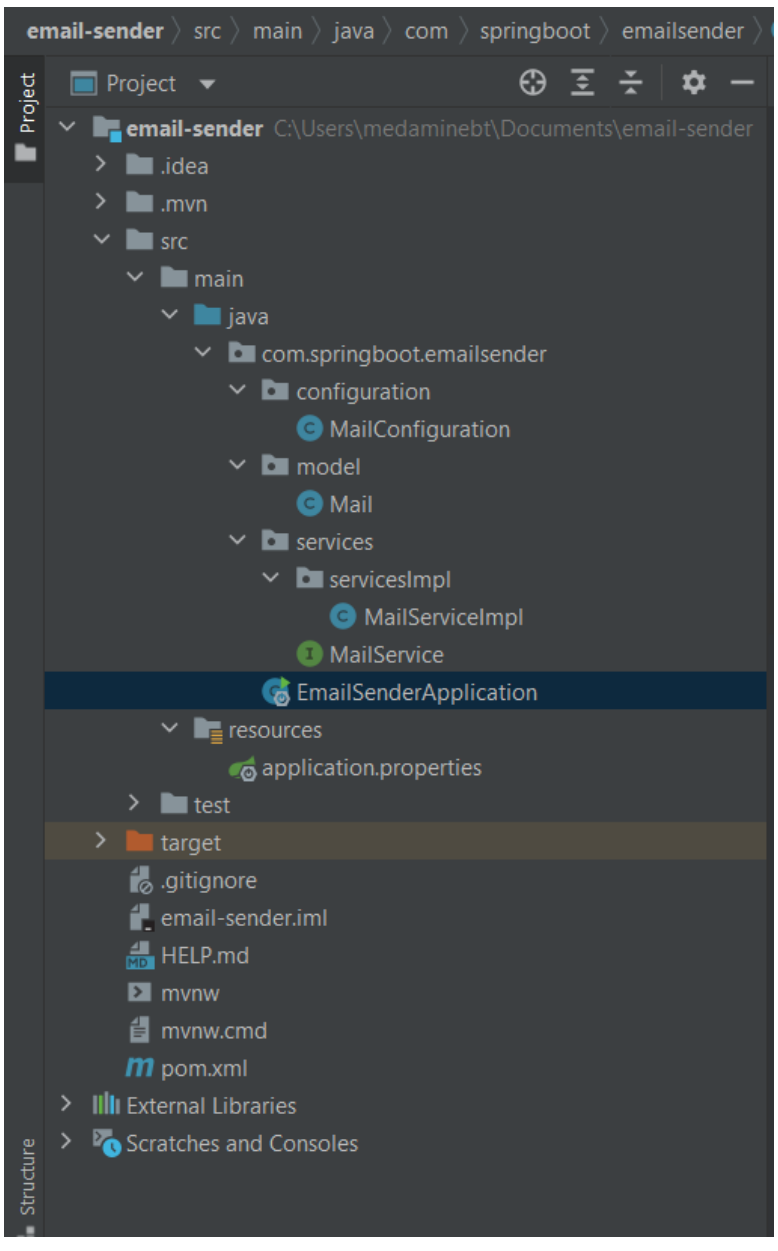
> ✔️ In this Tutorial we are going to be using **IntelliJ IDEA 2021.3.2** (any student can get this for free **here**)
>
> You can also use **IntelliJ community Edition** Or **STS** (spring tool suite) ***[you can use Spring Initializr https://start.spring.io/ to generate your project file]***

Setting up a Spring Boot Application





You can make architecture like this and it will still work (depends on the rest of your application architecture)



This is the architecture that we will be using - Intellij IDEA project

Tools and Technologies

1. IntelliJ IDEA Project
2. Maven
3. JDK 1.8 (Java 8)

4. Spring Boot `2.5.9`

Project Dependencies

We’re going to setup a simple Spring Boot application which demonstate how to send an email. Spring Boot provides auto-configuration for it as well as a starter module.

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

pom.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>2.5.9</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.springboot</groupId>
    <artifactId>email-sender</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>email-sender</name>
    <description>email-sender</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

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



    </dependencies>

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

</project>
```

Mail Service

Creates MailService interface and define sendEmail() method.

```
package com.springboot.emailsender.services;

import com.springboot.emailsender.model.Mail;

public interface MailService {
    public void sendEmail(Mail mail);
}
```

Mail Service Implementation

Mark MailServiceImpl class as "mailService" using `@Service` annotation.

Use `@Autowired` annotation to autowire JavaMailSender class.

javax.mail.internet.MimeMessage class represents a MIME style email message. Spring **MimeMessageHelper** class offers support for HTML text content, inline elements such as images, and typical mail attachments.

Spring **MimeMessageHelper.getMimeMessage()** return the populated underlying MimeMessage object along with attachments.

Finally use **JavaMailSender.send()** method to send populated MIME message.

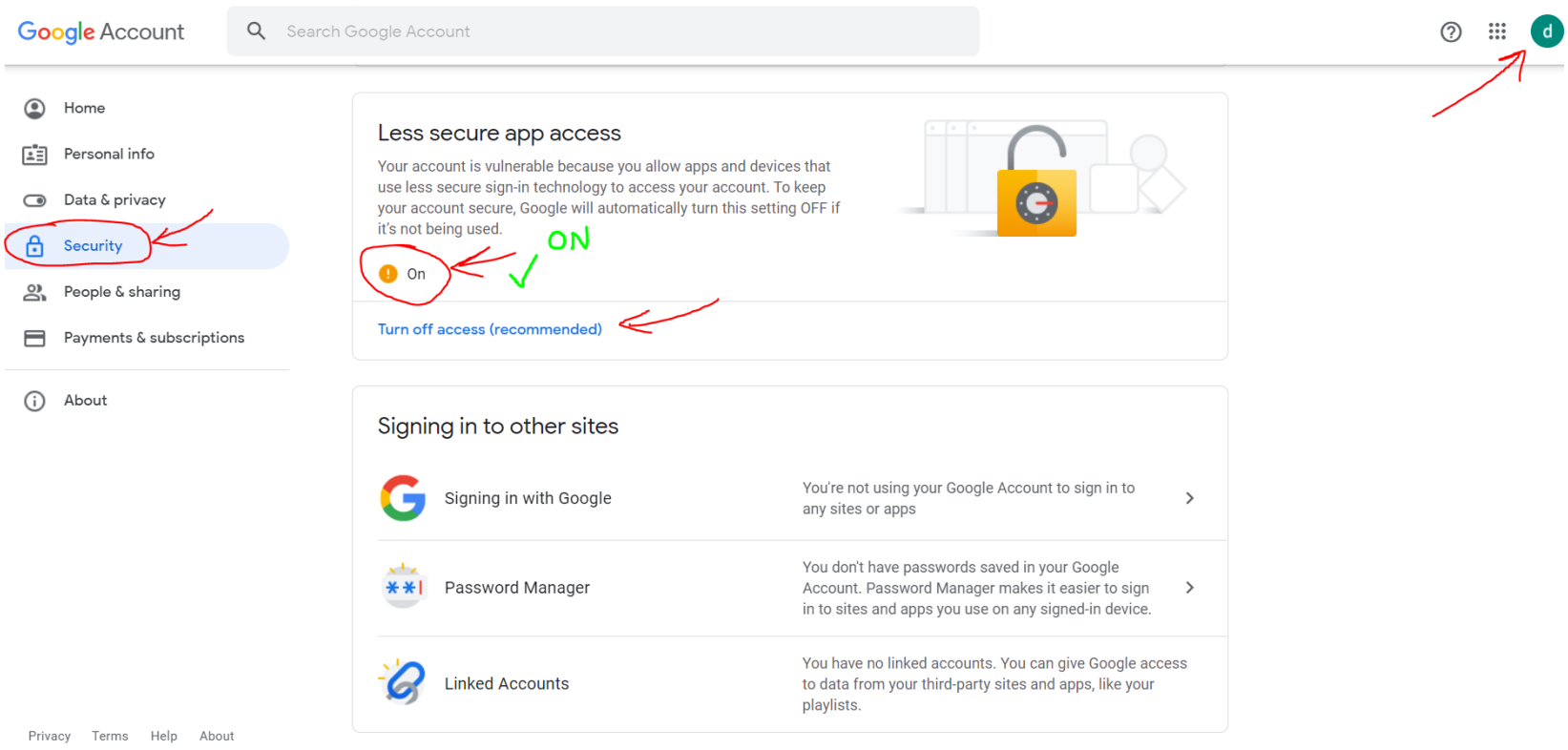
```
package com.springboot.emailsender.services.servicesImpl;

import java.io.UnsupportedEncodingException;
import javax.mail.MessagingException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import com.springboot.emailsender.services.MailService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import com.springboot.emailsender.model.Mail;

@Service("mailService")
public class MailServiceImpl implements MailService {

    @Autowired
    JavaMailSender mailSender;

    public void sendEmail(Mail mail) {
        MimeMessage mimeMessage = mailSender.createMimeMessage();

        try {

            MimeMessageHelper mimeMessageHelper = new MimeMessageHelper(mimeMessage, true);
            mimeMessageHelper.setSubject(mail.getMailSubject());
            mimeMessageHelper.setFrom(new InternetAddress(mail.getMailFrom(), "MailSenderAdmin name"));
            mimeMessageHelper.setTo(mail.getMailTo());
            mimeMessageHelper.setText(mail.getMailContent());

            mailSender.send(mimeMessageHelper.getMimeMessage());

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

Spring Boot Email Configuration

Define email configurations in **application.properties** file. I have used my gmail id to send an Email, You need to configure yours.

```
# Mail sender - springboot settings
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=yourGmailAddress@gmail.com
spring.mail.password=YourGmailPassword
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
mail.smtp.debug=true
```

Set up your gmail account settings to recieve the email correctly without the gmail security blocking the mail sender .



Gmail account settings

MailConfiguration Class

`@Configuration` annotation imports the Spring configuration. `@Configuration` objects are managed as Spring beans within the container, imported configurations are used to injected using @Autowired or @Inject.

Set Gmail SMTP configurations like host, port number, username and password, Which are defined in `application.properties` file.

```
package com.springboot.emailsender.configuration;

import java.util.Properties;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;


@Configuration
public class MailConfiguration {

    @Autowired
    private Environment env;

    @Bean
    public JavaMailSender getMailSender() {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

        mailSender.setHost(env.getProperty("spring.mail.host"));
        mailSender.setPort(Integer.valueOf(env.getProperty("spring.mail.port")));
        mailSender.setUsername(env.getProperty("spring.mail.username"));
        mailSender.setPassword(env.getProperty("spring.mail.password"));

        Properties javaMailProperties = new Properties();
        javaMailProperties.put("mail.smtp.starttls.enable", "true");
        javaMailProperties.put("mail.smtp.auth", "true");
        javaMailProperties.put("mail.transport.protocol", "smtp");
```

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

        mailSender.setJavaMailProperties(javaMailProperties);
        return mailSender;
    }
}
```

Mail Pojo

This is simple pojo class which contains different email attributes like mailFrom, mailTo, mailCc, mailBcc, mailSubject, mailContent, contentType and attachments.

```
package com.springboot.emailsender.model;

import java.util.Date;
import java.util.List;

public class Mail {
    private String mailFrom;

    private String mailTo;

    private String mailCc;

    private String mailBcc;

    private String mailSubject;

    private String mailContent;

    private String contentType;

    private List < Object > attachments;

    public Mail() {
        contentType = "text/plain";
    }

    public String getContentType() {
        return contentType;
    }

    public void setContentType(String contentType) {
        this.contentType = contentType;
    }

    public String getMailBcc() {
        return mailBcc;
    }

    public void setMailBcc(String mailBcc) {
        this.mailBcc = mailBcc;
    }

    public String getMailCc() {
        return mailCc;
    }

    public void setMailCc(String mailCc) {
        this.mailCc = mailCc;
    }

    public String getMailFrom() {
        return mailFrom;
    }

    public void setMailFrom(String mailFrom) {
        this.mailFrom = mailFrom;
    }

    public String getMailSubject() {
        return mailSubject;
    }

    public void setMailSubject(String mailSubject) {
        this.mailSubject = mailSubject;
    }
```

```
public String getMailTo() {
    return mailTo;
}

public void setMailTo(String mailTo) {
    this.mailTo = mailTo;
}

public Date getMailSendDate() {
    return new Date();
}

public String getMailContent() {
    return mailContent;
}

public void setMailContent(String mailContent) {
    this.mailContent = mailContent;
}

public List < Object > getAttachments() {
    return attachments;
}

public void setAttachments(List < Object > attachments) {
    this.attachments = attachments;
    }

}
```

Boot Strap class (Application.java)

Spring Boot Application class is used to bootstrap and launch a Spring application from a Java main method. This class automatically creates the ApplicationContext from the classpath, scan the configuration classes and launch the application.

```
package com.springboot.emailsender;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.ComponentScan;

import com.springboot.emailsender.model.Mail;
import com.springboot.emailsender.services.MailService;

@SpringBootApplication
@ComponentScan(basePackages = {
        "com.springboot"
})
public class EmailSenderApplication {

        public static void main(String[] args) {

            Mail mail = new Mail();
            mail.setMailFrom("YourGmailAccount@gmail.com");
            mail.setMailTo("YourGmailAccount@gmail.com");
            mail.setMailSubject("Spring Boot - Email Example");
            mail.setMailContent("Learn How to send Email using Spring Boot!!!\n\nThanks\nmabttech.medium.com");

            ApplicationContext ctx = SpringApplication.run(EmailSenderApplication.class, args);
            MailService mailService = (MailService) ctx.getBean("mailService");
            mailService.sendEmail(mail);

        }

        // SpringApplication.run(EmailSenderApplication.class, args);
    }
```

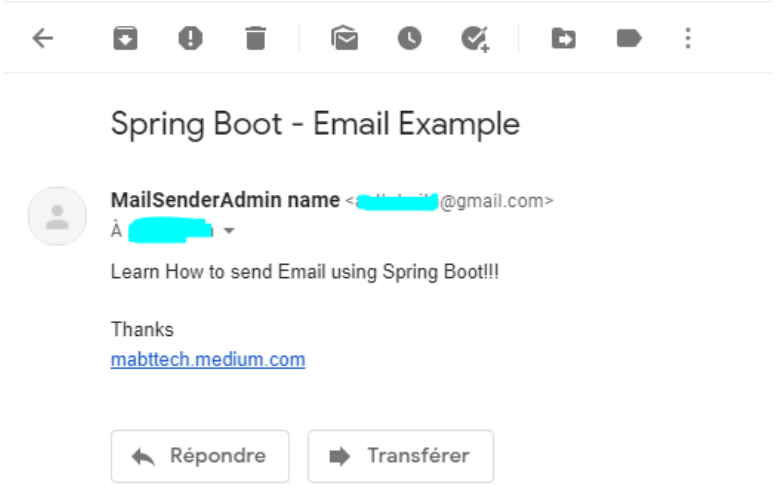
Run the Spring Boot Application

Run the Application.java class or You can use the `mvn spring-boot:run` command to run the spring boot application.

```
19:33:58.434 [Thread-1] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.
springframework.boot.devtools.restart.classloader.RestartClassLoader@36af82f0

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.9)

2022-02-19 19:33:58.700  INFO 32 --- [  restartedMain] c.s.emailsender.EmailSenderApplication   : Starting EmailSenderApplication using
 Java 1.8.0_60 on DESKTOP-39DASS0 with PID 32 (C:\Users\mabttech\Documents\email-sender\target\classes started by mabttech in C:\Users\m
abttech\Documents\email-sender)
2022-02-19 19:33:58.700  INFO 32 --- [  restartedMain] c.s.emailsender.EmailSenderApplication   : No active profile set, falling back to
default profiles: default
2022-02-19 19:33:58.715  INFO 32 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set
'spring.devtools.add-properties' to 'false' to disable
2022-02-19 19:33:59.055  INFO 32 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 3
5729
2022-02-19 19:33:59.070  INFO 32 --- [  restartedMain] c.s.emailsender.EmailSenderApplication   : Started EmailSenderApplication in 0.63
3 seconds (JVM running for 1.231)
DEBUG: Jakarta Mail version 1.6.7
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name: {com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTranspo
rt,Oracle], com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Oracle], com.sun.mail.pop3.POP
3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Oracle], com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSP
ORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Oracle], com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPS
tore,Oracle], com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Oracle]}
DEBUG: Providers Listed By Protocol: {imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Oracle], smtp=javax.mail.Provider
[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Oracle], pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Oracle], imaps=
javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Oracle], smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTP
SSLTransport,Oracle], pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Oracle]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
DEBUG: getProvider() returning javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Oracle]
DEBUG SMTP: useEhlo true, useAuth true
DEBUG SMTP: trying to connect to host "smtp.gmail.com", port 587, isSSL false
220 smtp.gmail.com ESMTP z17sm2951366wmf.11 - gsmtp
DEBUG SMTP: connected to host "smtp.gmail.com", port: 587
EHLO DESKTOP-39DASS0
250-smtp.gmail.com at your service, [197.244.98.146]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
DEBUG SMTP: Found extension "SIZE", arg "35882577"
DEBUG SMTP: Found extension "8BITMIME", arg ""
DEBUG SMTP: Found extension "STARTTLS", arg ""
DEBUG SMTP: Found extension "ENHANCEDSTATUSCODES", arg ""
DEBUG SMTP: Found extension "PIPELINING", arg ""
DEBUG SMTP: Found extension "CHUNKING", arg ""
DEBUG SMTP: Found extension "SMTPUTF8", arg ""
STARTTLS
220 2.0.0 Ready to start TLS
EHLO DESKTOP-39DASS0
250-smtp.gmail.com at your service, [197.244.98.146]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
DEBUG SMTP: Found extension "SIZE", arg "35882577"
DEBUG SMTP: Found extension "8BITMIME", arg ""
DEBUG SMTP: Found extension "AUTH", arg "LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH"
DEBUG SMTP: Found extension "ENHANCEDSTATUSCODES", arg ""
DEBUG SMTP: Found extension "PIPELINING", arg ""
DEBUG SMTP: Found extension "CHUNKING", arg ""
DEBUG SMTP: Found extension "SMTPUTF8", arg ""
DEBUG SMTP: protocolConnect login, host=smtp.gmail.com, user=YourGmailAccount@gmail.com, password=<non-null>
DEBUG SMTP: Attempt to authenticate using mechanisms: LOGIN PLAIN DIGEST-MD5 NTLM XOAUTH2
DEBUG SMTP: Using mechanism LOGIN
DEBUG SMTP: AUTH LOGIN command trace suppressed
DEBUG SMTP: AUTH LOGIN succeeded
DEBUG SMTP: use8bit false
MAIL FROM:<YourGmailAccount@gmail.com>
```

```
250 2.1.0 OK z17sm2951366wmf.11 - gsmtp
RCPT TO:<YourGmailAccount@gmail.com>
250 2.1.5 OK z17sm2951366wmf.11 - gsmtp
DEBUG SMTP: Verified Addresses
DEBUG SMTP:   YourGmailAccount@gmail.com
DATA
354  Go ahead z17sm2951366wmf.11 - gsmtp
Date: Sat, 19 Feb 2022 19:34:05 +0100 (GMT+01:00)
From: MailSenderAdmin name <YourGmailAccount@gmail.com>
To: YourGmailAccount@gmail.com
Message-ID: <1657043435.2.1645295649642@DESKTOP-39DASS0>
Subject: Spring Boot - Email Example
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="----=_Part_0_2145848273.1645295639086"

------=_Part_0_2145848273.1645295639086
Content-Type: multipart/related;
  boundary="----=_Part_1_414098156.1645295639086"

------=_Part_1_414098156.1645295639086
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Learn How to send Email using Spring Boot!!!

Thanks
mabttech.medium.com
------=_Part_1_414098156.1645295639086--

------=_Part_0_2145848273.1645295639086--
.
250 2.0.0 OK  1645295656 z17sm2951366wmf.11 - gsmtp
DEBUG SMTP: message successfully delivered to mail server
QUIT
221 2.0.0 closing connection z17sm2951366wmf.11 - gsmtp

Process finished with exit code 0
```

Email Output :



💡 **Email Sender - Spring boot projects / other tutos :**

https://www.tutorialspoint.com/spring_boot/spring_boot_sending_email.htm

https://www.technicalkeeda.com/spring-boot-tutorials/how-to-send-email-using-spring-boot

https://howtodoinjava.com/spring-boot2/send-email-with-attachment/

https://mkyong.com/spring-boot/spring-boot-how-to-send-email-via-smtp/

https://devstory.net/11145/spring-email

https://www.baeldung.com/spring-email