



INDEX

Spring Mail

1. Introduction [04](#)
2. Working with Email Message [05](#)

Spring Mail

Introduction

Java App ----- JDBC API -----> DB s/w

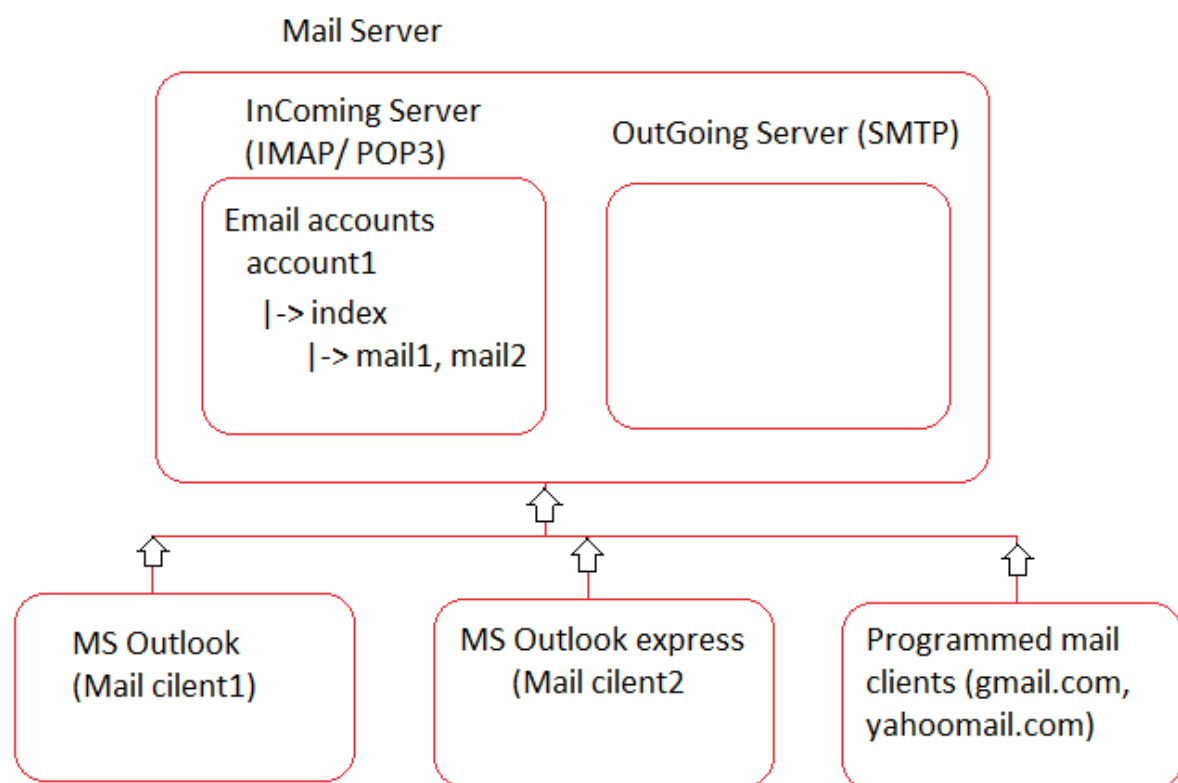
Java App ----- JNDI API -----> JNDI registry

Java App ----- Java Mail API -----> Mail server

(Manges emails accounts and email messages)

- James mail server
- MS exchange server
- SMTP One Server and etc.

- + Java Mail API is part of JEE module having packages (javax.mail, javax.mail.activation and etc.).
- + Spring Mail API provides abstraction on Java Mail API and simplifies email operations.



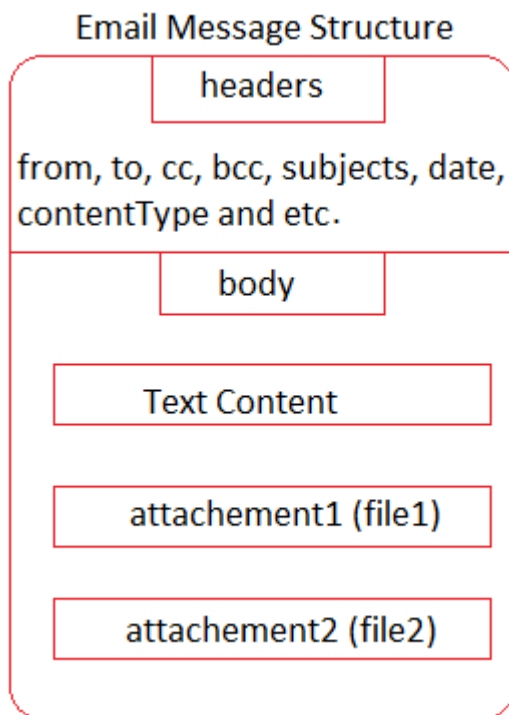
IMAP: Internet Mail Access Protocol

POP3: Post office protocol

SMTP: Simple Mail Transfer Protocol

- Outgoing servers are responsible for sending email messages.
- Incoming servers are responsible for receiving and holding email messages.

Working with Email Message



- Working with plain Java Mail API it takes more time and needs more complex code.
- Working with Spring Mail takes less time and needs simple code.
- If we add Spring Boot Mail starter to Spring boot project
 - a. Gives Java Mail related jar files.
 - b. Gives Spring beans like `JavaMailSenderImpl` class object through `AutoConfiguration`.
 - c. Simplifies the process of Mail Message creation having the attachment.

Note:

- ✓ Plain Java Mail API supports multiple mailing operations like send mail, read mail, delete mail, forward mail and etc.
- ✓ As of now Spring Mail is supporting only send mail operation to trigger email message the moment business Tx is completed.
- ✓ The moment add Spring Boot Mail starter jars to CLASSPATH and mail properties to `application.properties` the `JavaMailSenderImpl` object can be injected anywhere through Auto wiring process.

java mail properties

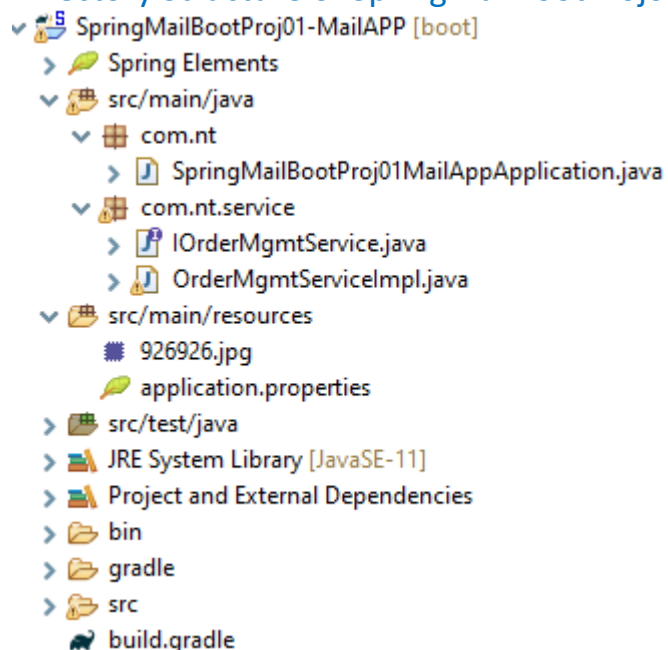
`spring.mail.host=smtp.gmail.com`
`spring.mail.port=587`

```
spring.mail.username=nirmalakumarsahu7@gmail.com
spring.mail.password=nimu@050599
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=5000
spring.mail.properties.mail.smtp.writetimeout=5000
```

- ✚ Based on these properties JavaMailSenderImpl class object will be created having connection with Gmail outgoing server and we can that object to create Mail Messages and to send email messages.

JavaMailSender (I)
| implements
JavaMailSendImpl (c)

Directory Structure of SpringMailBootProj01-MailAPP:



- Develop the above directory structure using Spring Starter Project option and create the package and classes also.
- Many jars dependencies will come automatically in build.gradle because while developing Spring Starter Project we have to choose the following jar.
 - Java Mail Sender
- Then use the following code with in their respective file.

IOrderMgmtService.java

```
package com.nt.service;

public interface IOrderMgmtService {
    public String purchase(String[] items, float[] prices, String
customerEmail, String[] cc, String[] bcc);
}
```

OrderMgmtServiceImpl.java

```
package com.nt.service;

import java.util.Arrays;
import java.util.Date;
import java.util.Random;

import javax.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;

@Service("orderService")
public class OrderMgmtServiceImpl implements IOrderMgmtService {

    @Autowired
    private JavaMailSender sender;

    @Override
    public String purchase(String[] items, float[] prices, String
customerEmail, String[] cc, String[] bcc) {
        // Calculate bill amount
        float bAmt = 0.0f;
        for (float p : prices)
            bAmt += p;
        String body = Arrays.toString(items) + "are purchased having
price " + Arrays.toString(prices)
            + " will bill amount " + bAmt + "rs. having order id "
+ new Random().nextInt(10000);
        boolean status = triggerMail(body, customerEmail, cc, bcc);
        return body + " mailed delivered: " + status;
    }
}
```

```

    private boolean triggerMail(String body, String customerEmail,
String[] cc, String[] bcc) {
        boolean status=false;
        try {
            //create MimeMessage object having using sender
object
            MimeMessage msg = sender.createMimeMessage();
            //create MimeMessageHelper class object to make
attachment process easy
            MimeMessageHelper helper = new
MimeMessageHelper(msg, true);
            //set content email message
            helper.setSubject("open to know it");
            helper.setTo(customerEmail);
            helper.setCc(cc);
            helper.setBcc(bcc);
            helper.setSentDate(new Date());
            helper.setText(body);
            helper.addAttachment("926926.jpg", new
ClassPathResource("926926.jpg"));
            sender.send(msg);
            status = true;
        } catch (Exception e) {
            e.printStackTrace();
            status=false;
        }
        return status;
    }
}

```

application.properties

```

# java mail properties
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=nirmalakumarsahu7@gmail.com
spring.mail.password=nimu34r55423
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.auth=true

```



```
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=5000
spring.mail.properties.mail.smtp.writetimeout=5000
```

application.properties

```
package com.nt;

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

import com.nt.service.IOrderMgmtService;

@SpringBootApplication
public class SpringMailBootProj01MailAppApplication {

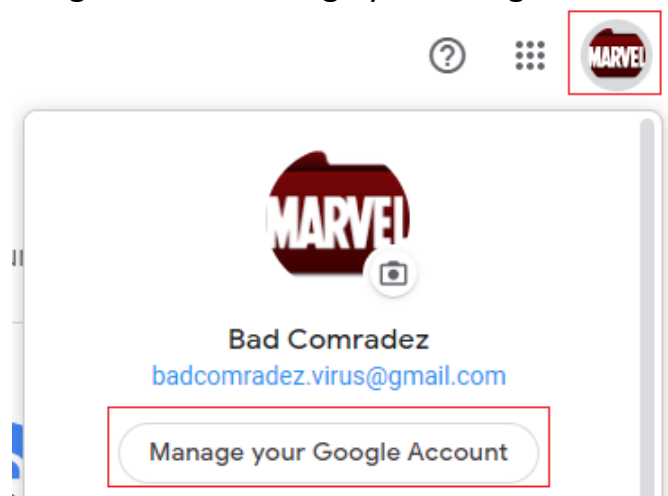
    public static void main(String[] args) {
        //get IoC container
        ApplicationContext ctx =
SpringApplication.run(SpringMailBootProj01MailAppApplication.class, args);
        //get service class object
        IOrderMgmtService service = ctx.getBean("orderService",
IOrderMgmtService.class);
        try {
            String result = service.purchase(new String[] {"clothes",
"crackers", "sweets", "dia"},

                                new float[] {5000, 4000, 3000,
1000},

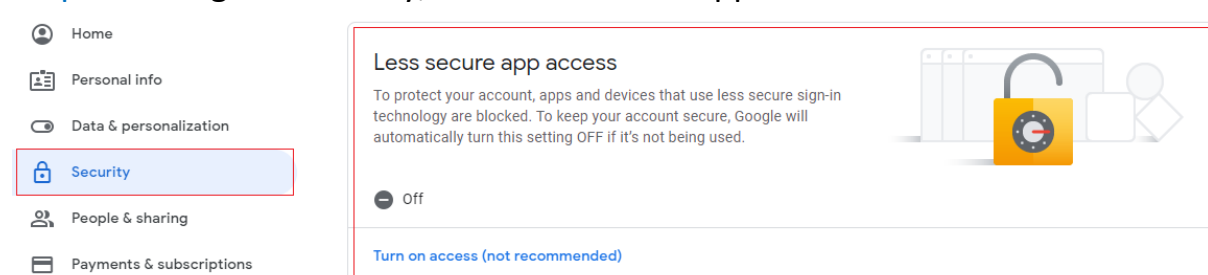
                                "badcomradez.virus@gmail.com",
                                new String[]
{"nirmalakumarsahu.official.99@gmail.com"},
                                new String[]
{"nirmalawebpractice@gmail.com"}));
            System.out.println(result);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Before running application, we have things to do the following things from sender's email account:

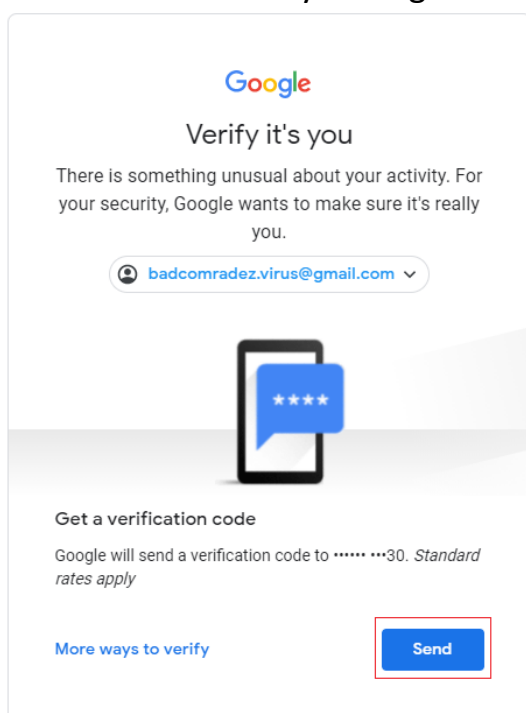
Step 1: Login to sender email account (like Gmail login), go to account letter/image click on Manage your Google Account.



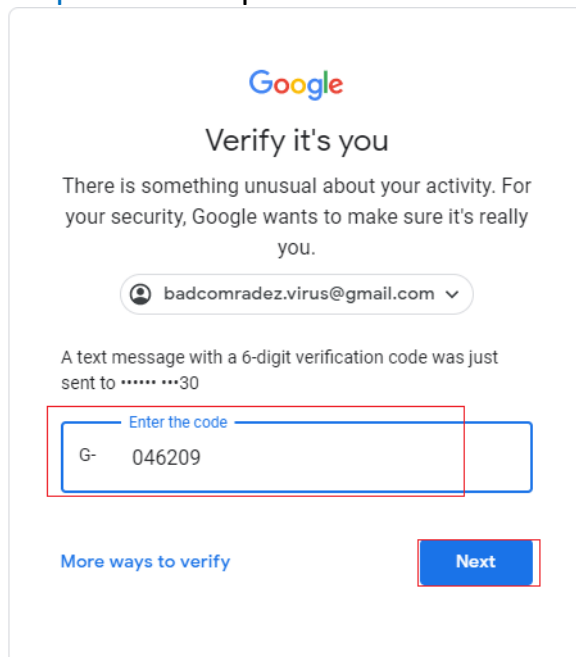
Step 2: Then go to Security, then Less secure app access click on that.



Step 3: Then it will open a verification that then click on send, it will send a verification code to your register mobile number.



Step 4: Then kept the received verification code then click on Next.

A screenshot of the Google account verification screen. At the top is the Google logo. Below it, the text "Verify it's you" is centered. A message follows: "There is something unusual about your activity. For your security, Google wants to make sure it's really you." Below this is a dropdown menu showing the email address "badcomradez.virus@gmail.com". A message states: "A text message with a 6-digit verification code was just sent to30". A text input field is shown with the placeholder "Enter the code" and the value "G- 046209". At the bottom left is a link "More ways to verify" and at the bottom right is a blue button labeled "Next".

Google

Verify it's you

There is something unusual about your activity. For your security, Google wants to make sure it's really you.

badcomradez.virus@gmail.com

A text message with a 6-digit verification code was just sent to30

Enter the code

G- 046209

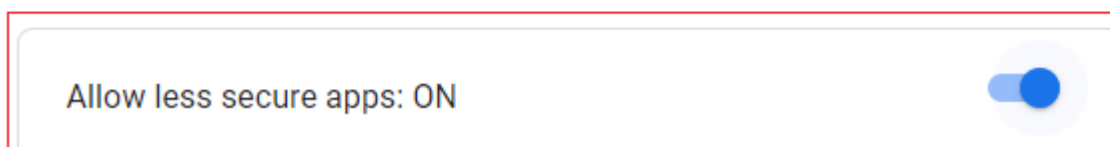
More ways to verify

Next

Step 5: Then ON the Allow less secure apps

← Less secure app access

Some apps and devices use less secure sign-in technology, which makes your account vulnerable. You can turn off access for these apps, which we recommend, or turn it on if you want to use them despite the risks. Google will automatically turn this setting OFF if it's not being used. [Learn more](#)

A screenshot of the "Allow less secure apps" setting. The text "Allow less secure apps: ON" is displayed on the left, and a blue toggle switch is on the right, indicating the setting is turned on.

Allow less secure apps: ON

----- The END -----