Java SMTP Email Send

2/2/2026

Transport send mail using java smtp

Here's a clean and working example of sending email using Java SMTP (JavaMail API) wit...

EMAIL SEDT mULTIPAL TO AND CC AND ATTACHMENT ALSO MULTIPALE

Here's a full production-ready Java SMTP example for: ✅ Multiple TO ✅ Multiple CC ✅...

Multipart multipart CAN I SEDN VIA REST API

Yes ✅ — you can absolutely send Multipart (attachments + data) via REST API and then...

Here's a clean and working example of **sending email using Java SMTP (JavaMail API)** with `Transport.send()` — exactly what you need for backend services, microservices, or Spring Boot apps.

1️⃣ Maven Dependency

```xml
<dependency>
    <groupId>com.sun.mail</groupId>
    <artifactId>jakarta.mail</artifactId>
    <version>2.0.1</version>
</dependency>
```

2️⃣ Basic SMTP Email Sender (Plain Java)
📧 Gmail SMTP Example

```java
import jakarta.mail.*;
import jakarta.mail.internet.*;
import java.util.Properties;

public class EmailSender {

    public static void main(String[] args) {

        final String username = "your_email@gmail.com";   // SMTP username
        final String password = "your_app_password";      // App password (not normal password)

        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");

        // Create session
        Session session = Session.getInstance(props, new Authenticator() {
```

```java
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(username, password);
        }
    });

    try {
        // Create message
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress(username));
        message.setRecipients(
                Message.RecipientType.TO,
                InternetAddress.parse("receiver@gmail.com")
        );
        message.setSubject("Test Mail from Java SMTP");
        message.setText("Hello 👋\nThis email is sent using Java SMTP Transport.send()");

        // Send mail
        Transport.send(message);

        System.out.println("✅ Email sent successfully");

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

3️⃣ HTML Email Version

java   Copy code

```java
message.setContent(
    "<h2>Hello Alpesh 👋</h2><p>This is <b>HTML Mail</b> using Java SMTP</p>",
    "text/html"
);
```

4️⃣ Attachment Example

java   Copy code

```java
MimeBodyPart textPart = new MimeBodyPart();
textPart.setText("Please find attachment");

MimeBodyPart filePart = new MimeBodyPart();
filePart.attachFile(new File("/mnt/data/report.pdf"));
```

```java
Multipart multipart = new MimeMultipart();
multipart.addBodyPart(textPart);
multipart.addBodyPart(filePart);

message.setContent(multipart);
```

5️⃣ Common SMTP Servers

Provider	Host	Port	TLS
Gmail	smtp.gmail.com	587	✅
Outlook	smtp.office365.com	587	✅
Yahoo	smtp.mail.yahoo.com	587	✅
AWS SES	email-smtp.ap-south-1.amazonaws.com	587	✅

🔐 Important (Gmail Users)

You must use **App Password**, not your Gmail password

👉 Google Account → Security → App Passwords

6️⃣ Spring Boot Version (Service Class)

java   Copy code

```java
@Service
public class SmtpMailService {

    public void sendMail(String to, String subject, String body) {

        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");

        Session session = Session.getInstance(props,
            new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication("your@gmail.com", "app_password");
                }
            });

        try {
```

```java
            Message msg = new MimeMessage(session);
            msg.setFrom(new InternetAddress("your@gmail.com"));
            msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));
            msg.setSubject(subject);
            msg.setText(body);

            Transport.send(msg);

        } catch (Exception e) {
            throw new RuntimeException("Mail send failed", e);
        }
    }
}
```

🧠 Architecture Tip (Microservice Design)

Since you mentioned earlier:

> *"I have a separate server only for sending mail (no DB, no AWS)"*

✅ Best practice design:

```css
Main Service → REST Call / Queue → Mail Service → SMTP Server
```

or

```css
Main Service → SQS / Kafka → Mail Microservice → SMTP
```

This is **enterprise-grade architecture** (scalable + fault tolerant).

If you want, I can give you:
✅ SMTP with **S3 attachment streaming**
✅ SMTP with **SQS trigger**
✅ SMTP with **Async sending**
✅ SMTP Mail Microservice full project structure
✅ Retry + DLQ logic
✅ Non-blocking mail sending using virtual threads (Java 21+)

Just tell me what architecture you're building 😄

EMAIL SEDT mULTIPAL TO AND CC AND ATTACHMENT ALSO MULTIPALE

Here's a **full production-ready Java SMTP example** for:

✅ Multiple **TO**
✅ Multiple **CC**
✅ Multiple **Attachments**
✅ HTML body
✅ Uses `Transport.send()`
✅ Enterprise-style structure

📧 Java SMTP – Multiple TO, CC, Attachments

Maven Dependency

```xml
<dependency>
    <groupId>com.sun.mail</groupId>
    <artifactId>jakarta.mail</artifactId>
    <version>2.0.1</version>
</dependency>
```

✅ Complete Working Code

```java
import jakarta.mail.*;
import jakarta.mail.internet.*;
import java.io.File;
import java.util.*;

public class MultiMailSender {

    public static void main(String[] args) {

        final String SMTP_USER = "your_email@gmail.com";
        final String SMTP_PASS = "your_app_password"; // Gmail App Password

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

```java
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.port", "587");

Session session = Session.getInstance(props, new Authenticator() {
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(SMTP_USER, SMTP_PASS);
    }
});

try {
    Message message = new MimeMessage(session);

    // FROM
    message.setFrom(new InternetAddress(SMTP_USER));

    // ---------- MULTIPLE TO ----------
    String[] toEmails = {
        "user1@gmail.com",
        "user2@gmail.com",
        "user3@gmail.com"
    };

    InternetAddress[] toAddresses = new InternetAddress[toEmails.length];
    for (int i = 0; i < toEmails.length; i++) {
        toAddresses[i] = new InternetAddress(toEmails[i]);
    }
    message.setRecipients(Message.RecipientType.TO, toAddresses);

    // ---------- MULTIPLE CC ----------
    String[] ccEmails = {
        "cc1@gmail.com",
        "cc2@gmail.com"
    };

    InternetAddress[] ccAddresses = new InternetAddress[ccEmails.length];
    for (int i = 0; i < ccEmails.length; i++) {
        ccAddresses[i] = new InternetAddress(ccEmails[i]);
    }
    message.setRecipients(Message.RecipientType.CC, ccAddresses);

    // SUBJECT
    message.setSubject("Multi TO + CC + Multiple Attachments Test");

    // ---------- BODY ----------
    MimeBodyPart htmlPart = new MimeBodyPart();
    htmlPart.setContent(
        "<h2>Hello Team 👋</h2>" +
        "<p>This mail contains:</p>" +
        "<ul>" +
```

```java
                "<li>Multiple TO</li>" +
                "<li>Multiple CC</li>" +
                "<li>Multiple Attachments</li>" +
                "</ul>",
                "text/html"
            );

            // ---------- ATTACHMENTS ----------
            List<String> filePaths = List.of(
                "/mnt/data/report1.pdf",
                "/mnt/data/image1.png",
                "/mnt/data/data.xlsx"
            );

            Multipart multipart = new MimeMultipart();
            multipart.addBodyPart(htmlPart); // add body first

            for (String path : filePaths) {
                MimeBodyPart filePart = new MimeBodyPart();
                filePart.attachFile(new File(path));
                multipart.addBodyPart(filePart);
            }

            // SET CONTENT
            message.setContent(multipart);

            // ---------- SEND ----------
            Transport.send(message);

            System.out.println("✅ Email sent successfully to multiple users with CC and
attachments");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

🧠 Important Notes

Multiple TO

```java
java                                                                      Copy code

message.setRecipients(Message.RecipientType.TO, toAddresses);
```

Multiple CC

```java
message.setRecipients(Message.RecipientType.CC, ccAddresses);
```

Multiple Attachments

```java
for (String path : filePaths) {
    MimeBodyPart filePart = new MimeBodyPart();
    filePart.attachFile(new File(path));
    multipart.addBodyPart(filePart);
}
```

🏗 Microservice Architecture (Correct Design)

Since you already said you have a **separate mail server**:

```css
Main Service → REST / SQS / Kafka → Mail Service → SMTP Server
```

This code fits perfectly inside your **Mail Microservice**.

⚡ Production Enhancements (Recommended)

If you're building real systems:

✅ Async sending (`@Async` or ExecutorService)
✅ Retry mechanism
✅ DLQ queue
✅ SMTP connection pool
✅ Template engine (Thymeleaf/Freemarker)
✅ Queue-based sending (SQS/Kafka/RabbitMQ)
✅ Streaming attachments (S3 → SMTP without disk)

If you want, I can give you a **ready-made Mail Microservice project structure**:

```arduino
mail-service/
├ controller
├ service
```

```
├ dto
├ config
├ queue-listener
├ retry-handler
├ smtp-client
```

with:
✅ REST API
✅ JSON request
✅ Multiple TO/CC/BCC
✅ Multiple attachments
✅ S3 streaming
✅ Queue-based sending
✅ Async + retry
✅ Logging + tracing

Just say: **"Give full mail microservice project"** 😄

Multipart multipart CAN I SEDN VIA REST API

Yes ✅ — **you can absolutely send `Multipart` (attachments + data) via REST API** and then convert it into `MimeMultipart` for SMTP email sending.

This is a very common **mail-microservice architecture** pattern.

🧩 Architecture Flow

```scss
Client (Postman / Frontend / Service)
        ↓
REST API (multipart/form-data)
        ↓
Mail Microservice
        ↓
Convert to MimeMultipart
        ↓
SMTP (Transport.send)
```

✅ REST API → SMTP with Multipart

1 REST Controller (Spring Boot)

```java
@RestController
@RequestMapping("/api/mail")
public class MailController {

    @Autowired
    private MailService mailService;

    @PostMapping(value = "/send", consumes = MediaType.MULTIPART_FORM_DATA_VALUE)
    public ResponseEntity<String> sendMail(
            @RequestParam("to") List<String> to,
            @RequestParam(value = "cc", required = false) List<String> cc,
            @RequestParam("subject") String subject,
            @RequestParam("body") String body,
            @RequestPart(value = "files", required = false) List<MultipartFile> files
    ) {
        mailService.sendMail(to, cc, subject, body, files);
        return ResponseEntity.ok("Mail sent successfully ✅");
    }
}
```

2 Mail Service (Convert REST Multipart → SMTP MimeMultipart)

```java
@Service
public class MailService {

    private final String SMTP_USER = "your@gmail.com";
    private final String SMTP_PASS = "app_password";

    public void sendMail(List<String> to,
                         List<String> cc,
                         String subject,
                         String body,
                         List<MultipartFile> files) {

        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");

        Session session = Session.getInstance(props, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
```

```java
                return new PasswordAuthentication(SMTP_USER, SMTP_PASS);
            }
        });

        try {
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(SMTP_USER));

            // TO
            message.setRecipients(Message.RecipientType.TO,
                    to.stream().map(t -> {
                        try { return new InternetAddress(t); }
                        catch (Exception e) { throw new RuntimeException(e); }
                    }).toArray(InternetAddress[]::new)
            );

            // CC
            if (cc != null && !cc.isEmpty()) {
                message.setRecipients(Message.RecipientType.CC,
                        cc.stream().map(c -> {
                            try { return new InternetAddress(c); }
                            catch (Exception e) { throw new RuntimeException(e); }
                        }).toArray(InternetAddress[]::new)
                );
            }

            message.setSubject(subject);

            // BODY
            MimeBodyPart bodyPart = new MimeBodyPart();
            bodyPart.setContent(body, "text/html");

            Multipart multipart = new MimeMultipart();
            multipart.addBodyPart(bodyPart);

            // ATTACHMENTS
            if (files != null) {
                for (MultipartFile file : files) {
                    MimeBodyPart filePart = new MimeBodyPart();
                    filePart.setFileName(file.getOriginalFilename());
                    filePart.setDataHandler(
                        new DataHandler(new ByteArrayDataSource(file.getBytes(),
file.getContentType())))
                    );
                    multipart.addBodyPart(filePart);
                }
            }

            message.setContent(multipart);
```

```
            Transport.send(message);

        } catch (Exception e) {
            throw new RuntimeException("Mail send failed", e);
        }
    }
}
```

🧪 Postman Test

Method: `POST`

URL: `http://localhost:8080/api/mail/send`

Body → form-data:

Key	Type	Value
to	text	user1@gmail.com
to	text	user2@gmail.com
cc	text	cc1@gmail.com
subject	text	Test Mail
body	text	`<h1>Hello</h1><p>REST multipart mail</p>`
files	file	report.pdf
files	file	image.png
files	file	data.xlsx

🔥 This is Enterprise Pattern

This is exactly how:

- Gmail APIs work
- AWS SES API works
- Enterprise mail services work
- Notification microservices work

🧠 Key Concept

REST Multipart ≠ SMTP Multipart

You convert:

Copy code

```lua
REST MultipartFile     --->   MimeBodyPart
List<MultipartFile> --->   MimeMultipart
```

⚡ Advanced (Production-grade)

If you're building real infra:
✅ Async sending
✅ Queue-based (SQS/Kafka)
✅ Retry + DLQ
✅ Streaming S3 → SMTP (no disk)
✅ Template engine
✅ Non-blocking virtual threads
✅ Circuit breaker
✅ Bulk email batching

If you want, I can give you a **full Mail Microservice architecture** with:

```arduino
                                        Copy code
Controller
DTO
Service
SMTP Client
Queue Listener
Retry Engine
DLQ Handler
Async Executor
Config
Logger
```

and JSON + multipart support both.

Just say: **"Full mail microservice project"** 🚀