ROHAN THAPA
thaparohan2019@gmail.com

Sending Email in Spring Boot

Why do we need to send emails?

Sending emails in applications serves multiple crucial purposes:

- **User Communication:** Emails are a direct channel to communicate with users, providing important information, updates, and notifications.
- **Security:** Emails are used for verification processes such as account creation, password resets, and two-factor authentication (2FA). This ensures that only authorized users can access their accounts.
- **Marketing and Engagement:** Emails help in promoting products, services, or updates, keeping users engaged with the platform.

Why do we need to send emails?

- **Transaction Receipts and Alerts:** In e-commerce or financial applications, emails are used to send transaction receipts, order confirmations, or alerts about suspicious activity.
- **User Experience:** Email notifications enhance user experience by keeping them informed about their interactions with the application, such as order status, event reminders, or social activity.

Overview

- **Email in Spring Boot:** Spring Boot simplifies sending emails using the **JavaMailSender** interface, which is built on top of the **JavaMail API**.

- **Generating OTP:** We'll generate a random four-digit OTP.

- **Sending Email:** We'll configure Spring Boot to send an email using Ethereal, a fake SMTP service useful for testing.

- **Implementation:** We'll write the code for generating the OTP, sending the email, and testing the service.

Internal Working

- **JavaMailSender:** This interface abstracts the complexity of JavaMail API and simplifies sending emails in Spring.

- **SMTP Configuration:** The SMTP server details in application.properties allow Spring Boot to connect to the email server.

- **Ethereal:** Acts as a fake SMTP server, useful for testing without sending real emails.

Add Dependencies

First, add the required dependencies to your **pom.xml** (for Maven) or **build.gradle** (for Gradle) file.

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

Configure Spring Boot

Configure your application to use a **SMTP server (Ethereal for testing)**. Add the following properties in **application.properties** or **application.yml**:

```
spring.mail.host=smtp.ethereal.email
spring.mail.port=587
spring.mail.username=barry.weber94@ethereal.email
spring.mail.password=RjDa5YFy3JXn3mYm3x
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.starttls.enabled=true
```

Configure Spring Boot

Explanation:

- **spring.mail.host:** SMTP server address.
- **spring.mail.port:** SMTP server port.
- **spring.mail.username/password:** Credentials for SMTP authentication.
- **mail.smtp.auth** and **mail.smtp.starttls.enable:** Ensure secure email transmission.

Generating OTP

We'll create a generateOTP methods to generate a six-digit OTP.

```
private String generateOtp(){
    SecureRandom secureRandom = new SecureRandom();
    int otp = secureRandom.nextInt(900000) + 100000;
    return String.valueOf(otp);
}
```

Creating Email Service

Now, we'll create a service to send otp via email using

```java
@Service
public class SendMailService {

    @Autowired
    private JavaMailSender javaMailSender;

    @Value("${spring.mail.username}")
    private String fromEmailId;

    public void sendMail(String receipt , String body, String subject){
        SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
        simpleMailMessage.setFrom(fromEmailId);
        simpleMailMessage.setTo(receipt);
        simpleMailMessage.setText(body);
        simpleMailMessage.setSubject(subject);

        javaMailSender.send(simpleMailMessage);
    }

}
```

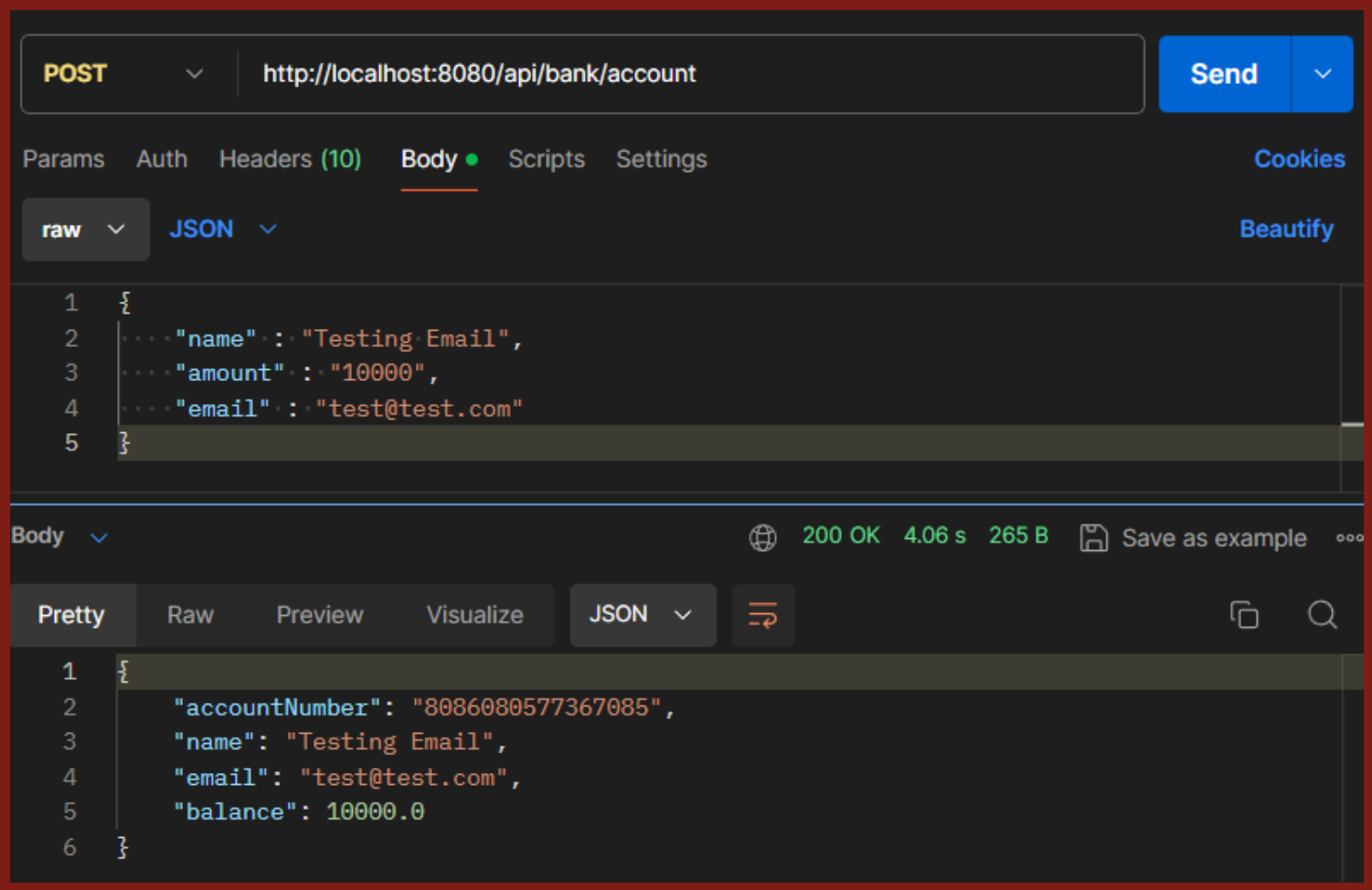
Sending Email On Account Creation

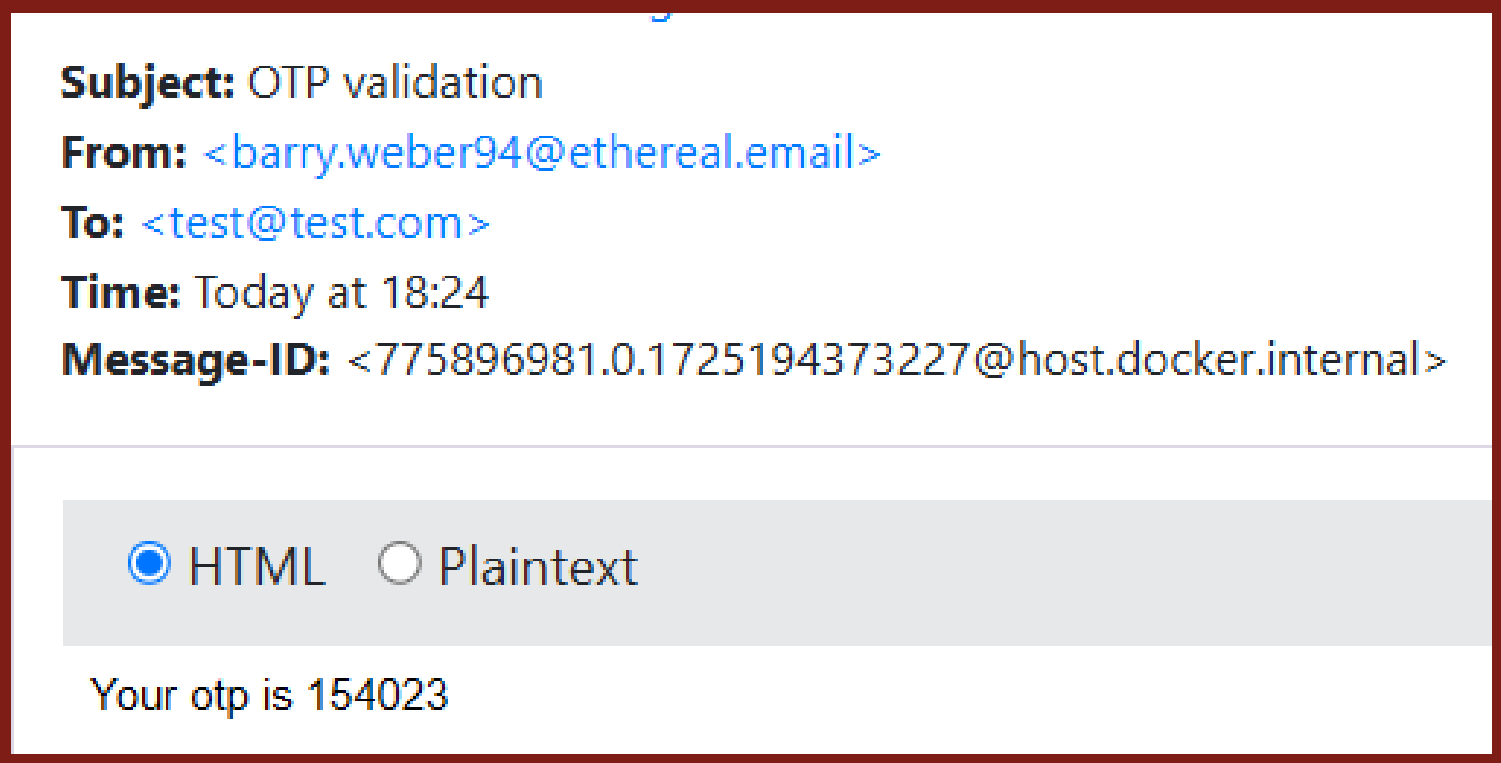
Now, we'll send email on account creation

```
public Account createAccount(CreateAccount createAccount) {
    String accountNumber = generateAccount();
    Account account = new Account(
        accountNumber,
        createAccount.getName(),
        createAccount.getEmail(),
        createAccount.getAmount());
    String body = "Your otp is " + generateOtp();
    sendMailService.sendMail(createAccount.getEmail(),body,"OTP validation");
    return accountRepository.save(account);
}
```

Output

Testing with Etherial Email





Advanced Concepts

- **Asynchronous Email Sending:** Use **@Async** on the **sendOtpEmail** method to send emails asynchronously.

- **Custom Email Templates:** Instead of **SimpleMailMessage**, you can use **MimeMessage** for sending HTML emails with **Thymeleaf** or **FreeMarker templates**.

Sending Email With Attachment

Why Include Attachments in Emails?

Attachments in emails are often used for:

1. **Sharing Documents:** Sending reports, contracts, invoices, or any other document that needs to be reviewed or signed.
2. **Multimedia Content:** Including images, videos, or audio files for marketing, tutorials, or presentations.
3. **Supporting Information:** Providing additional information or resources related to the email's content.

Sending Email With Attachment

To send an email with an attachment, we need to use MimeMessage instead of SimpleMailMessage.

```java
@Service
public class SendMailAttachmentService {

    @Autowired
    private JavaMailSender javaMailSender;

    @Value("${spring.mail.username}")
    private String fromEmailId;

    public void sendMail(String receipt , String body, String subject) throws MessagingException {

        String attachmentPath = "src/main/resources/static/image.png";

        MimeMessage mimeMessage = javaMailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(mimeMessage,true);
        helper.setFrom(fromEmailId);
        helper.setTo(receipt);
        helper.setText(body);
        helper.setSubject(subject);

        FileSystemResource file = new FileSystemResource(new File(attachmentPath));
        if (file.exists()) {
            helper.addAttachment("image.png", file);
            System.out.println("Attachment added successfully.");
        } else {
            System.out.println("Attachment file not found.");
        }

        javaMailSender.send(mimeMessage);
    }

}
```

Sending Email With Attachment

Modify the controller to use the new method that sends the attachment.

```
public Account createAccount(CreateAccount createAccount) throws MessagingException {
    String accountNumber = generateAccount();
    Account account = new Account(
        accountNumber,
        createAccount.getName(),
        createAccount.getEmail(),
        createAccount.getAmount());
    String body = "Your otp is " + generateOtp();
    sendMailAttachmentService.sendMail(createAccount.getEmail(),body,"OTP validation");
    return accountRepository.save(account);
}
```

Output

POST http://localhost:8080/api/bank/account Send

Params Auth Headers (10) Body Scripts Settings Cookies

raw JSON Beautify

```
{
    "name" : "Testing Email With Attachment",
    "amount" : "50000",
    "email" : "image@test.com"
}
```

Body 200 OK 5.14 s 282 B Save as example

Pretty Raw Preview Visualize JSON

```
{
    "accountNumber": "9777501065722510",
    "name": "Testing Email With Attachment",
    "email": "image@test.com",
    "balance": 50000.0
}
```

Public URL of this message

image.png

Subject: OTP validation
From: <barry.weber94@ethereal.email>
To: <image@test.com>
Time: Today at 19:01
Message-ID: <938889423.2.1725196569094@host.docker.internal>

HTML Plaintext

Your otp is 966903

Internal Working

- **MimeMessage:** A more complex email message format that supports **attachments, inline content, and multipart messages**. It is essential for emails that include anything beyond simple text.

- **MimeMessageHelper:** A helper class that simplifies the process of creating a **MimeMessage**. It allows setting **recipients, subjects**, and **content,** and **attaching files** all in one go.

- **FileSystemResource:** A Spring utility that wraps a file on the **filesystem**, making it easy to attach it to an email.

Thank You

ROHAN THAPA
thaparohan2019@gmail.com