Sending Emails with Java

Last updated: January 8, 2024

Written by: baeldung
Reviewed by: Michal Aibin

Java

1. Overview

In this quick tutorial, we're going to look at sending an email with and without attachments using the core Java mail library.

2. Project Setup and Dependency

For this article, we'll be using a simple Maven-based project with a dependency on Angus Mail. This is the Eclipse implementation of the Jakarta Mail API specification:

```
<dependency>
    <groupId>org.eclipse.angus</groupId>
    <artifactId>angus-mail</artifactId>
    <version>2.0.1</version>
</dependency>
```

The latest version can be found here.

3. Sending a Plain Text and an HTML Email

First, we need to configure the library with our email service provider's credentials. Then we'll create a *Session* that'll be used in constructing our message for sending.

The configuration is via a Java *Properties* object:

```
Properties prop = new Properties();
prop.put("mail.smtp.auth", true);
prop.put("mail.smtp.starttls.enable", "true");
prop.put("mail.smtp.host", "smtp.mailtrap.io");
prop.put("mail.smtp.port", "25");
prop.put("mail.smtp.ssl.trust", "smtp.mailtrap.io");
```

In the properties configuration above, we configured the email host as Mailtrap and used the port provided by the service as well.

Now let's create a session with our username and password:

```
Session session = Session.getInstance(prop, new Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(username, password);
    }
});
```

The username and password are given by the mail service provider alongside the host and port parameters.

Now that we have a mail *Session* object, let's create a *MimeMessage* for sending:

```
Message message = new MimeMessage(session);
message.setFrom(new InternetAddress("from@gmail.com"));
message.setRecipients(
  Message.RecipientType.TO, InternetAddress.parse("to@gmail.com"));
message.setSubject("Mail Subject");

String msg = "This is my first email using JavaMailer";

MimeBodyPart mimeBodyPart = new MimeBodyPart();
mimeBodyPart.setContent(msg, "text/html; charset=utf-8");

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

message.setContent(multipart);

Transport.send(message);
```

In the snippet above, we first created a *message* instance with the necessary properties — to, from and subject. This is followed by a *mimeBodyPart* that has an encoding of *text/html* since our message is styled in HTML.

Next, we created an instance of *MimeMultipart* object that we can use to wrap the *mimeBodyPart* we created.

Finally, we set the *multipart* object as the content of our *message* and used the *send()* of *Transport* object to do the mail sending.

So, we can say that the *mimeBodyPart* is contained in the *multipart* that is contained in the *message*. This way, a *multipart* can contain more than one *mimeBodyPart*.

This is going to be the focus of the next section.

4. Sending Email With an Attachment

Next, to send an attachment, we only need to create another *MimeBodyPart* and attach the file(s) to it:


```
MimeBodyPart attachmentBodyPart = new MimeBodyPart();
attachmentBodyPart.attachFile(new File("path/to/file"));
```

We can then add the new body part to the *MimeMultipart* object we created earlier:

```
multipart.addBodyPart(attachmentBodyPart);
```

That's all we need to do.

Once again, we set the *multipart* instance as the content of the *message* object, and finally we'll use the *send()* to do the mail sending.

5. Formatting Email Text

To format and style our email text, we can use HTML and CSS tags.

For example, if we want our text to be bold, we will implement the *<b>* tag. For coloring the text, we can use the *style* tag. **We can also combine HTML tags with CSS tags if we want to have additional properties, such as bold.**

Let's create a *String* containing bold-red text:

```
String msgStyled = "This is my <b style='color:red;'>bold-red email</b> using JavaMailer";
```

This *String* will hold our styled text to be sent in the email body.

6. Conclusion

In this article, we've seen how to use the Jakarta Mail API to send emails even with attachments.

As always, the complete source code is available over on GitHub.

Comments are closed on this article!