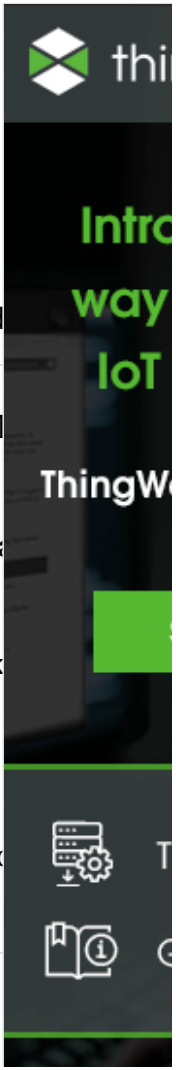
Send, read Emails and Appointments From MS Exchange [using Java]

This code will help you connect your app to MS Exchange using the EWS Java API. In its current form, the fetches emails and appointments and can be broadened.

by Shantanu Sikdar · Dec. 27, 16 · Java Zone



While enabling my application with MS Exchange, I end up writing this code. I used the EWS Java API (EWSJavaAPI_1.2.jar) (The JAR I upl this snippet.).

The snippet fetches emails and appointments with limited features, like subjects, sender name, appointment times, etc. But we can very well components and make it a full-fledged email application. Feel free to add your components.

I've tested the code against Exchange Version 2010 and 2007 and it worked well. But firs, please check to see if EWS is enabled for the MS Ex are using.

Also, if any of you readers have something better to suggest, please do so.

It's been quite long I revisited this post of mine. In the original version, I, unfortunately, missed the email sending code. That's been remedie can find the email sending code in the snippet below.

```
1  import microsoft.exchange.webservices.data.*;
2
3  import java.net.URI;
```

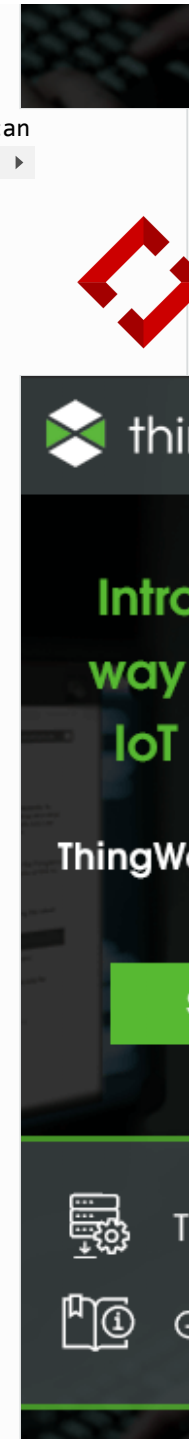
```
import java.util.*;

/**
 * @author Shantanu Sikdar
 */

public class MSExchangeEmailService {

    private static ExchangeService service;
    private static Integer NUMBER_EMAILS_FETCH = 5; // only latest 5 emails/appointments are fetched.

    /**
     * Firstly check, whether "https://webmail.xxxx.com/ews/Services.wsdl" and "https://webmail.xxxx.com/ews/Exchange.asmx"
     * is accessible, if yes that means the Exchange Webservice is enabled on your MS Exchange.
     */
    static {
        try {
            service = new ExchangeService(ExchangeVersion.Exchange2010_SP1);
//service = new ExchangeService(ExchangeVersion.Exchange2007_SP1); //depending on the version of your Exchange.
            service.setUrl(new URI("https://webmail.xxxx.com/ews/Exchange.asmx"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Initialize the Exchange Credentials.
     * Don't forget to replace the "USRNAME","PWD","DOMAIN_NAME" variables.
     */
    public MSExchangeEmailService() {
        ExchangeCredentials credentials = new WebCredentials("USRNAME", "PWD", "DOMAIN_NAME");
        service.setCredentials(credentials);
```

```
    }

    /**
     * Reading one email at a time. Using Item ID of the email.
     * Creating a message data map as a return value.
     */
    public Map readEmailItem(ItemId itemId) {
        Map messageData = new HashMap();
        try {
            Item itm = Item.bind(service, itemId, PropertySet.FirstClassProperties);
            EmailMessage emailMessage = EmailMessage.bind(service, itm.getId());
            messageData.put("emailItemId", emailMessage.getId().toString());
            messageData.put("subject", emailMessage.getSubject().toString());
            messageData.put("fromAddress", emailMessage.getFrom().getAddress().toString());
            messageData.put("senderName", emailMessage.getSender().getName().toString());
            Date dateTimeCreated = emailMessage.getDateTimeCreated();
            messageData.put("SendDate", dateTimeCreated.toString());
            Date dateTimeRecieved = emailMessage.getDateTimeReceived();
            messageData.put("RecievedDate", dateTimeRecieved.toString());
            messageData.put("Size", emailMessage.getSize() + "");
            messageData.put("emailBody", emailMessage.getBody().toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return messageData;
    }

/**
 * Number of email we want to read is defined as NUMBER_EMAILS_FETCH,
 */
    public List>
```

```
68      readEmails() {
69          List > msgDataList = new ArrayList > ();
70          try {
71              Folder folder = Folder.bind(service, WellKnownFolderName.Inbox);
72              FindItemsResults results = service.findItems(folder.getId(), new ItemView(NUMBER_EMAILS_FETCH));
73              int i = 1;
74              for (Item item : results) {
75                  Map messageData = new HashMap();
76                  messageData = readEmailItem(item.getId());
77                  System.out.println("\nEmails #" + (i++) + ":");
78                  System.out.println("subject : " + messageData.get("subject").toString());
79                  System.out.println("Sender : " + messageData.get("senderName").toString());
80                  msgDataList.add(messageData);
81              }
82          } catch (Exception e) {
83              e.printStackTrace();
84          }
85          return msgDataList;
86      }
87
88      /**
89       * Reading one appointment at a time. Using Appointment ID of the email.
90       * Creating a message data map as a return value.
91       */
92      public Map readAppointment(Appointment appointment) {
93          Map appointmentData = new HashMap();
94          try {
95              appointmentData.put("appointmentItemId", appointment.getId().toString());
96              appointmentData.put("appointmentSubject", appointment.getSubject());
97              appointmentData.put("appointmentStartTime", appointment.getStart() + "");
98              appointmentData.put("appointmentEndTime", appointment.getEnd() + "");
99              //appointmentData.put("appointmentBody", appointment.getBody().toString());
```

```
        } catch (ServiceLocalException e) {
            e.printStackTrace();
        }
        return appointmentData;
    }

    /**
     *Number of Appointments we want to read is defined as NUMBER_EMAILS_FETCH,
     *  Here I also considered the start data and end date which is a 30 day span.
     *  We need to set the CalendarView property depending upon the need of ours.
     */
    public List>

    readAppointments() {
        List > apntmtDataList = new ArrayList > ();
        Calendar now = Calendar.getInstance();
        Date startDate = Calendar.getInstance().getTime();
        now.add(Calendar.DATE, 30);
        Date endDate = now.getTime();
        try {
```

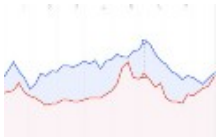
```
            CalendarFolder calendarFolder = CalendarFolder.bind(service, WellKnownFolderName.Calendar, new PropertySet());

            CalendarView cView = new CalendarView(startDate, endDate, 5);
            cView.setPropertySet(new PropertySet(AppointmentSchema.Subject, AppointmentSchema.Start, AppointmentSchema.End));// we can

            // as well depending upon our need.

            FindItemsResults appointments = calendarFolder.findAppointments(cView);

            int i = 1;

            List appList = appointments.getItems();

            for (Appointment appointment : appList) {

                System.out.println("\nAPPOINTMENT #" + (i++) + ":");

                Map appointmentData = new HashMap();

                appointmentData = readAppointment(appointment);

                System.out.println("subject : " + appointmentData.get("appointmentSubject").toString());

                System.out.println("On : " + appointmentData.get("appointmentStartTime").toString());

                apntmtDataList.add(appointmentData);

            }
        } catch (Exception e) {

            e.printStackTrace();

        }

        return apntmtDataList;

    }
```



```
    public void sendEmails(List recipientsList) {

        try {

            StringBuilder strBldr = new StringBuilder();

            strBldr.append("The client submitted the SendAndSaveCopy request at:");

            strBldr.append(Calendar.getInstance().getTime().toString() + " .");

            strBldr.append("Thanks and Regards");

            strBldr.append("Shantanu Sikdar");

            EmailMessage message = new EmailMessage(service);

            message.setSubject("Test sending email");

            message.setBody(new MessageBody(strBldr.toString()));

            for (String string : recipientsList) {

                message.getToRecipients().add(string);

            }

            message.sendAndSaveCopy();

        } catch (Exception e) {

            e.printStackTrace();

        }

        System.out.println("message sent");

    }
```

```
    public static void main(String[] args) {

        MSExchangeEmailService msees = new MSExchangeEmailService();

        msees.readEmails();

        msees.readAppointments();

        List recipientsList = new ArrayList<>();

        recipientsList.add("email.id1@domain1.com");

        recipientsList.add("email.id2@domain1.com");

        recipientsList.add("email.id3@domain2.com");

        msees.sendEmails(recipientsList);

    }

}
```


Like This Article? Read More From DZone



Runtime Metrics in Execution Plans

Convert Table Data to List Using Reflection

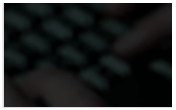
How to Send Emails Using Swift Mailer

Free DZone Refcard

Getting Started With Scala

Topics: JAVA, EMAIL, MS EXCHANGE, EWS, TUTORIAL

Topics: JAVA , EMAIL , MS EXCHANGE , EWS , TUTORIAL



Opinions expressed by DZone contributors are their own.

Java Partner Resources

Building Reactive Microservices in Java: Asynchronous and Event-Based Application Design
Red Hat Developer Program

NoSQL Options for Java Developers | Okta Developer
Okta

jQuery UI and Auto-Complete Address Entry
Melissa Data

Modern Java EE Design Patterns: Building Scalable Architecture for Sustainable Enterprise Development
Red Hat Developer Program