Javamail API

Example javamail can be browsed at https://github.com/apache/tomee/tree/master/examples/javamail

This is just a simple example to demonstrate a very basic usage of the API. It should be enough to get you started using the java mail packages.

The Code

A simple REST service using the Javamail API

Here we see a very simple RESTful endpoint that can be called with a message to send by Email. It would not be hard to modify the application to provide more useful configuration options. As is, this will not send anything, but if you change the parameters to match your mail server then you'll see the message being sent. You can find much more detailed information on the Javamail API here

```
package org.superbiz.rest;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import java.util.Date;
import java.util.Properties;

@Path("/email")
public class EmailService {

    @POST
    public String lowerCase(final String message) {

        try {

            //Create some properties and get the default Session
            final Properties props = new Properties();
            props.put("mail.smtp.host", "your.mailserver.host");
            props.put("mail.debug", "true");

            final Session session = Session.getInstance(props, new Authenticator() {
                @Override
```

```
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication("MyUsername", "MyPassword");
                }
            });

            //Set this just to see some internal logging
            session.setDebug(true);

            //Create a message
            final MimeMessage msg = new MimeMessage(session);
            msg.setFrom(new InternetAddress("your@email.address"));
            final InternetAddress[] address = {new InternetAddress(
"general@tomitribe.com")};
            msg.setRecipients(Message.RecipientType.TO, address);
            msg.setSubject("JavaMail API test");
            msg.setSentDate(new Date());
            msg.setText(message, "UTF-8");


            Transport.send(msg);
        } catch (MessagingException e) {
            return "Failed to send message: " + e.getMessage();
        }

        return "Sent";
    }
}
```

Testing

Test for the JAXRS service

The test uses the OpenEJB ApplicationComposer to make it trivial.

The idea is first to activate the jaxrs services. This is done using @EnableServices annotation.

Then we create on the fly the application simply returning an object representing the web.xml. Here we simply use it to define the context root but you can use it to define your REST Application too. And to complete the application definition we add @Classes annotation to define the set of classes to use in this app.

Finally to test it we use cxf client API to call the REST service post() method.

```
package org.superbiz.rest;

import org.apache.cxf.jaxrs.client.WebClient;
import org.apache.openejb.jee.WebApp;
import org.apache.openejb.junit.ApplicationComposer;
import org.apache.openejb.testing.Classes;
import org.apache.openejb.testing.EnableServices;
import org.apache.openejb.testing.Module;
import org.junit.Test;
import org.junit.runner.RunWith;

import java.io.IOException;

import static org.junit.Assert.assertEquals;

@EnableServices(value = "jaxrs")
@RunWith(ApplicationComposer.class)
public class EmailServiceTest {

    @Module
    @Classes(EmailService.class)
    public WebApp app() {
        return new WebApp().contextRoot("test");
    }

    @Test
    public void post() throws IOException {
        final String message = WebClient.create("http://localhost:4204").path(
"/test/email/").post("Hello Tomitribe", String.class);
        assertEquals("Failed to send message: Unknown SMTP host: your.mailserver.host
", message);
    }
}
```

Running

Running the example is fairly simple. In the "javamail-api" directory run:

```
$ mvn clean install
```

Which should create output like the following.

```
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to create
one for the beans deployed.
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
```

```
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager,
provider-id=Default Transaction Manager)
INFO - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Initializing network services
INFO - Creating ServerService(id=cxf-rs)
INFO - Creating ServerService(id=httpejbd)
INFO - Created ServicePool 'httpejbd' with (10) core threads, limited to (200) threads
with a queue of (9)
INFO - Initializing network services
INFO -   ** Bound Services **
INFO -   NAME                 IP              PORT
INFO -   httpejbd             127.0.0.1       4204
INFO - -------
INFO - Ready!
INFO - Configuring enterprise application:
D:\github\tomee\examples\javamail\EmailServiceTest
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.rest.EmailServiceTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Creating Container(id=Default Managed Container)
INFO - Using directory D:\windows\tmp for stateful session passivation
INFO - Configuring Service(id=comp/DefaultManagedExecutorService, type=Resource,
provider-id=Default Executor Service)
INFO - Auto-creating a Resource with id 'comp/DefaultManagedExecutorService' of type
'javax.enterprise.concurrent.ManagedExecutorService for 'test'.
INFO - Configuring Service(id=comp/DefaultManagedScheduledExecutorService,
type=Resource, provider-id=Default Scheduled Executor Service)
INFO - Auto-creating a Resource with id 'comp/DefaultManagedScheduledExecutorService'
of type 'javax.enterprise.concurrent.ManagedScheduledExecutorService for 'test'.
INFO - Configuring Service(id=comp/DefaultManagedThreadFactory, type=Resource,
provider-id=Default Managed Thread Factory)
INFO - Auto-creating a Resource with id 'comp/DefaultManagedThreadFactory' of type
'javax.enterprise.concurrent.ManagedThreadFactory for 'test'.
INFO - Enterprise application "D:\github\tomee\examples\javamail\EmailServiceTest"
loaded.
INFO - Creating dedicated application classloader for EmailServiceTest
INFO - Assembling app: D:\github\tomee\examples\javamail\EmailServiceTest
INFO - Using providers:
INFO -      org.apache.johnzon.jaxrs.JohnzonProvider@2687f956
INFO -      org.apache.cxf.jaxrs.provider.JAXBElementProvider@1ded7b14
INFO -      org.apache.johnzon.jaxrs.JsrProvider@29be7749
INFO -      org.apache.johnzon.jaxrs.WadlDocumentMessageBodyWriter@5f84abe8
INFO -      org.apache.openejb.server.cxf.rs.EJBAccessExceptionMapper@4650a407
INFO -      org.apache.cxf.jaxrs.validation.ValidationExceptionMapper@30135202
INFO - REST Application: http://127.0.0.1:4204/test/       ->
org.apache.openejb.server.rest.InternalApplication
INFO -      Service URI: http://127.0.0.1:4204/test/email  -> Pojo
org.superbiz.rest.EmailService
INFO -              POST http://127.0.0.1:4204/test/email/ ->      String
```

```
lowerCase(String)
INFO - Deployed Application(path=D:\github\tomee\examples\javamail\EmailServiceTest)
DEBUG: JavaMail version 1.4ea
DEBUG: java.io.FileNotFoundException: D:\java\jdk8\jre\lib\javamail.providers (The
system cannot find the file specified)
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.providers
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name:
{com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.s
mtp.SMTPSSLTransport,Sun Microsystems, Inc],
com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.S
MTPTransport,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPS
SLStore,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3S
SLStore,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore
,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store
,Sun Microsystems, Inc]}
DEBUG: Providers Listed By Protocol:
{imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun
Microsystems, Inc],
imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun Microsystems,
Inc], smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun
Microsystems, Inc],
pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems,
Inc], pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun
Microsystems, Inc],
smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun
Microsystems, Inc]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.address.map
DEBUG: java.io.FileNotFoundException: D:\java\jdk8\jre\lib\javamail.address.map (The
system cannot find the file specified)
DEBUG: setDebug: JavaMail version 1.4ea
DEBUG: getProvider() returning
javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems,
Inc]
DEBUG SMTP: useEhlo true, useAuth false
DEBUG SMTP: trying to connect to host "your.mailserver.host", port 25, isSSL false
INFO - Undeploying app: D:\github\tomee\examples\javamail\EmailServiceTest
INFO - Stopping network services
INFO - Stopping server services
```