



MIND STORM SOFTWARE PVT LTD

Google App Engine Training



ExamResults – Step 5 – Email Service

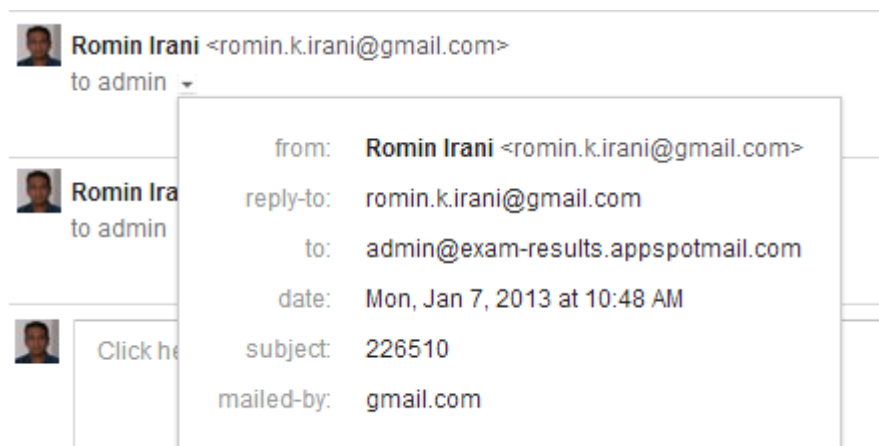
In this session, we are going to build the Email Layer for the application. In the previous session, we build the XMPP layer for the ExamResults application that allowed us to interact with our application via XMPP clients like Google Chat.

So what we plan to do here is the following:

- 1) Allow a user to interact with the Exam Results app by sending an Email to it.
- 2) The Email subject will contain the Seat Number of the student.
- 3) Our application will receive the email, parse out the Seat Number from the Email Subject, get the results from the database and send back an email reply with the results.

ExamResults – Step 5 – See it in Action

Shown below is an example of an email that was sent to the live application hosted at <http://exam-results.appspot.com>



The email was sent to admin@exam-results.appspotmail.com with the **subject** containing the *seat number* of the student.

The App Engine application responded in a while with the Email reply that contained the Exam Result as shown below:

Your Exam Results



Inbox x



admin@exam-results.appspotmail.com via [2uix4h7xygsz66weerlq.apphosting.l](#)
to me ▾

Seat Number : 226510

Student Name : Romin Irani

Mathematics : 60

Communication Skills : 70

Electronic Circuits : 90

Programming Languages : 80

Total Marks : 300/400

Percentage : 75%

=====

Thank you for using the Exam Helper Bot

Refer to **hands-on-exercises/Exam Results App/step 5** for full project source code. You can also import it directly into Eclipse and study the code.

Import the Project

We will import the Eclipse project at this stage so that you have all the working source code for this step.

1. In Eclipse, go to **File → Import**. This will bring up the Import dialog.
2. Select **General → Existing Projects into Workspace** and click on **Next**.
3. For Select root directory, click on browse and go to **C:\appengine-training\hands-on-exercises\Exam Results App\step 5**
4. Ensure that **Copy projects into workspace** is selected.
5. Click on **Finish**.

Add service entry to appengine-web.xml

Since are going to make use of the Email Service, we have to indicate to App Engine that we will be using that service by adding the service to the **WEB-INF/appengine-web.xml** file.

We have added the following entry to the file, anywhere inside of the root element.

```
<inbound-services>
  <service>email</service>
</inbound-services>
```

If you already have an **<inbound-services>** element present in your **appengine-web.xml** file, you can simply add the **<service>email</service>** element inside it.

Add the Email Servlet (ExamResultsEmailHandlerServlet.java)

We will write a new Servlet **ExamResultsEmailHandlerServlet.java** that will process the incoming Email Message, execute the required functionality and send back a response as an Email reply.

When your App Engine application receives an Email Message, the App Engine infrastructure packages the Email Message into a HTTP Request so that it is easy for you to extract out the data and then do the processing depending on the text that has been sent.

Take a look at the source code for the **ExamResultsEmailHandlerServlet.java**. The entire source code is present in **hands-on-exercises/Exam Results App/step 5/src/com/mindstormsoftware/examresults/ExamResultsEmailHandlerServlet.java** file

Let us go through the code step by step:

- 1) The first thing that we do is get wrap the incoming HTTP Request into a Mail MimeMessage object as shown below:

```
MimeMessage message = new MimeMessage(session, req.getInputStream());
```

- 2) Once we have the MimeMessage object, we can inspect it by calling methods like `getContentType()`, `getSubject()` and also get the sender Address by calling the `getFrom()`.
- 3) The `getSubject()` method will return us the text that is present there. We are expecting it to contain the seat number of student for which the exam results are required.
- 4) We simply invoke the **ExamResultDAO.INSTANCE.getExamResult(seatNumber)** to retrieve the Exam Results and we form a reply message in plain text format.
- 5) We then compose the reply MimeMessage using setting the From, Recipient, Subject and Text of the Email reply message.
- 6) Finally, we use the Mail Transport class to send the message to App Engine and ask it to deliver it to the recipient.

Add Servlet entry to web.xml

We need to add the servlet entry to **web.xml**, so that our application is aware of the Servlet. Remember that all Email Messages to your application will be sent on the following web path **/_ah/mail/***, so we will map our Servlet to that, so that whenever an email message is sent, our servlet is invoked.

The following servlet entries are added to **web.xml**

```
<!-- ExamResults Incoming Email Handler -->
<servlet>
    <servlet-name>ExamResultsEmailHandler</servlet-name>
    <servlet-
class>com.mindstormsoftware.examresults.ExamResultsEmailHandlerServlet</servlet-
class>
</servlet>
<servlet-mapping>
    <servlet-name>ExamResultsEmailHandler</servlet-name>
    <url-pattern>/_ah/mail/*</url-pattern>
```

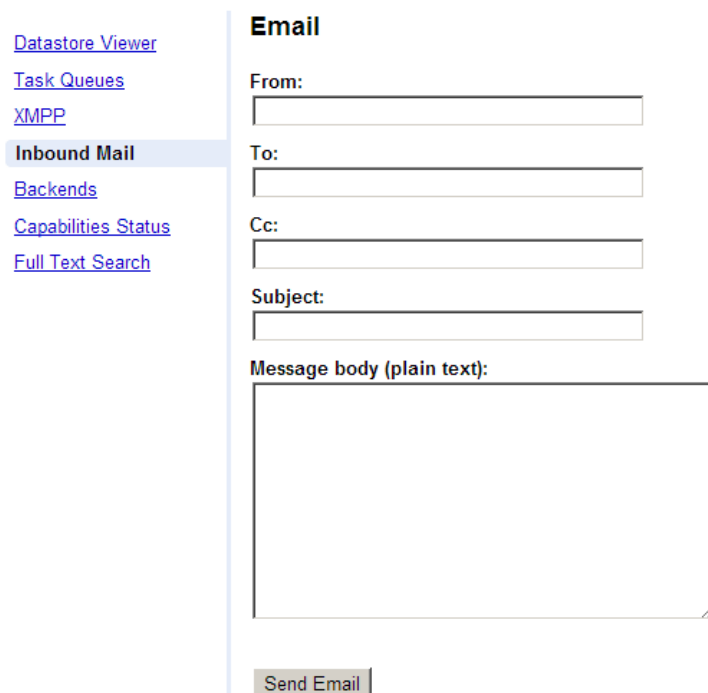
```
</servlet-mapping>
```

Instructions to Test locally

Once you made the changes, test out the changes locally first. Stop the Web Application if it is already running by clicking on the Stop icon in the Console window. Restart the application i.e. Right-click on Project and select **Run As → Web Application** and navigate to <http://localhost:8888>

You can test out the Email interactions locally via the Admin console that is available on http://localhost:8888/_ah/admin url.

Go to the above url and click on the link **Inbound Email** as shown below:



The screenshot shows the 'Inbound Email' form in the Admin console. On the left is a sidebar with links: [Datastore Viewer](#), [Task Queues](#), [XMPP](#), [Inbound Mail](#) (highlighted), [Backends](#), [Capabilities Status](#), and [Full Text Search](#). The main area is titled 'Email' and contains the following fields:

- From:** A text input field.
- To:** A text input field.
- Cc:** A text input field.
- Subject:** A text input field.
- Message body (plain text):** A large text area.
- Send Email:** A button at the bottom right of the form.

You can now send a message to your application by populating the fields as given below:

- **From:** Select as Chat message
- **To:** Provide an email address over here. It can be anything.
- **Subject:** Provide an email address here. It can be anything in the local environment.
- **Message Body (plain text):** Just put a space or something here. It does not matter since we are extracting out only the Email subject but the tool here requires a value, so just enter some value here.
- Click on **Send Email**. Sample screenshot is shown below:
- This will send the message to our running application. In the Admin console, you will just see a message sent successfully notification as shown below:

Message has been sent at 05:58:41 India Standard Time

- If you switch to your Eclipse and view the Console, you will see that the email message was received successfully and that the message is sent out by our Application. You will just see the text and no email actually gets sent out by the Development Server, but this validates that the Email functionality is working. A screenshot of the console is shown below:

```
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO: MailService.send
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:   From: admin@exam-results.appspotmail.com
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:   To: test@test.com
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:   Reply-to: admin@exam-results.appspotmail.com
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:   Subject: Your Exam Results
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:   Body:
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:     Content-type: text/plain
Jan 8, 2013 12:28:41 AM com.google.appengine.api.mail.dev.LocalMailService log
INFO:     Data length: 266
```

Instructions to Deploy and Test Live

To deploy to the cloud, right-click on project and then **Google → Deploy to App Engine**.

Once the application is deployed, you can use any email client (Gmail, Outlook) to send an email to it.

Remember that your application is live at <http://<appid>.appspot.com> , so you need to send an email to **<anything>@<appid>.appspotmail.com**

Once the email is received, you will get an email reply within some time.

Next Session

In the next session, we shall see how to add Cron Service support to the application. A Cron Service allows us to write repetitive tasks that could be executed at defined intervals. E.g. send an email every 12 hours with latest status, take backup every Wednesday at 3:00 PM and so on.