FAQ

Questions

General

Design-related

Actions

Platform Specific

Apple

Answers

General

Can I use OpenJDK and ESME together?

Don't use OpenJDK and ESME together. YUI compressor fails with OpenJDK.

Why can't you delete messages in ESME?

There is a long discussion of the dangers of deleting messages on the mailing list.

On which J2EE containers has ESME been tested?

- Tomcat
- Jetty
- SAP Netweaver CE
- Glassfish

How does search work in ESME?

ESME uses Compassto perform full-text searches.

Selecting different usage patterns

There are different ways to use compass in ESME. The selection of which approach to use depends on what your requirements are.

To select the option that you wish to use, you must set this option in your property file. To set the option, please use the following syntax:

```
compass_config_file=/props/compass.filesystem.cfg.xml
```

Currently, there are three examples of different compass based options in the */server/src/main/resources /props/ folder.*

File name	Description
compass.filesystem.cfg. xml	Using the file system to store index related data
compass.jdbc.cfg.xml	Using the a database to store index related data - access via JDBC
compass.jndi.cfg.xml	Using the a database to store index related data - access via JNDI

How do you configure OpenID?

Using a proxy with OpenID

Often you may wish to install ESME behind the firewall. In most cases, you won't have an internal OpenId provider, which means that your users are going to use OpenIds from external providers. Usually, you will have to go through an internal proxy.

If this is the case, then you must set the proxy server. To do this use the property file, This file must include the following lines:

```
http.proxyHost=[proxy - look in your browser]
http.proxyPort=[proxy port - look in your browser]
```

Does ESME support LDAP integration?

Currently, ESME doesn't support LDAP usage. We are looking at a new LDAP component for liftas the basis for our implementation.

Does ESME work with JDK 1.5?

Currently, there are some issues dealing with JDK 1.5. Many of these issues are related to restrictions associated with Scala itself.

How do you start jetty with a different port when using maven?

mvn jetty:run -Djetty.port=8081

How do you run ESME in production-mode?

Start the container with the following environment variable (from lift list) *-Drun.mode=production*

How do you run ESME with CURL?

Step-0: Generate Token:_Login to ESME via WebGui and used the Tokens menu at the top to create a new token.

_Step-1:Construct the curl command:_Here is my example-_curl -X POST -v -u [your user id]:[your token from step-0] -d status="my curl message on feb 24"http://localhost:8080/esme/twitter/statuses/update.xml

My ESME has 'esme' context running in Jetty. I didn't make any config change and is the default deployment. My WebGUI login is,http://localhost:8080/esme/

Step-2: Execute curl command:
Make sure you have curl installed on your system. My system is Ubuntu. Cut n paste the curl command in the terminal and hit enter. (For convenience you can keep your WebGui running to check the message). After you hit enter, you should get 200 ok status from the server and a XML response. You can check the new message in the WebGUI and it updates REAL TIME! (no refresh needed)

Run multiple times curl to see the updates in WebGui. I used Firefox.

Design-related

How does Twitter's new streaming API differ from ESME's design?

Twitter now has another take on a message-streaming API over HTTP, using what looks like a non-HTTP-1.1-compliant form of request pipelining (sending multiple responses over a single open connection). See the documentation at http://apiwiki.twitter.com/Streaming-API-Documentation

The advantage of their mechanism is that it's a smoother experience. What we've done with chunking/long polling is to simulate a stream of data on top of a non-streaming protocol. What Twitter has done is to say "this is a one-way conversation, we've got an open TCP/IP connection, so let's use it." Implementing what they have would require going below the current set of abstractions that Lift provides above the Servlets. At a practical level, the difference is at one layer... the one dealing with the HTTP requests. At the layers above, events flow either way.

At the basic philosophical level, Twitter's implementation is purer. It treats a stream of information as a stream of information.
I like it, but I'm not sure what the benefits would be vs. the development costs of implementing such a mechanism (unless there's an en mass migration of microblogging clients to such a mechanism).

A significant disadvantage of Twitter's design is the requirement of only one streaming connection per account. As much as I dislike the approach of using session cookies to uniquely identify API message queues, it is a heck of a lot better than what is going to happen when Twitter clients start to implement this API, which will be:
1. I log in with Seesmic Web (which has implemented the Twitter streaming API)
2. I receive messages 1, 2, and 3.
3. I log in on a different computer with Twhirl (which has also
implemented the Twitter streaming API)
(3.1 Twitter disconnects the Seesmic connection invisibly from the user)
4. I receive message 4 in Twhirl
(4.1 Seesmic tries to reconnect, which results in Twhirl being disconnected)
5. I receive message 5 in Seesmic
6. And so on....
End result: 1 really confused user trying to connect from two banned IP addresses.
I think this is a good illustration of why we need some client-specific identifier for a streaming/delta-queue API. It doesn't need to be a session, but that's working pretty nicely for now.
I would prefer to stick with what Lift provides for the moment. I need to do the conceptual exercise, but on first glance I don't think Twitter's approach results in much of a gain over our approach. Fewer connection attempts, which will help a lot at Twitter-scale, but which I'm not sure makes a big difference at Enterprise-scale.
Another drawback (and I'm really not sure on this one) is that I don't think a lot of HTTP client libraries give easy access to a request that is still open. The design of the queue API is extremely simple from a client programming perspective. I think that's a big upside.

Question: In an enterprise context it could be an requirement to send a link to someone else pointing to a specific potentially old message in a certain Pool.

Yes. That's perfectly reasonable. That message is like a static file on disk. Once it's written, it remains unchanged until it's deleted. This is an ideal application of a REST-style approach. That's why I've advocated
for a "message based" approach first, but a REST/static approach when the message based approach doesn't make sense. What I am opposed to is a "try to make everything fit the REST model" approach to API design.

Question: Would it be costly in your model to get the message nr. X (+ n older messages) in a users timeline?
A message will exist outside of a timeline. There exists a cache of recently accessed messages. Sometimes there will be a historic message that is referenced and that will be materialized from backing store and rendered. It will likely fall out of cache if it's historical and not accessed again.

Question: I don't get why it has to be in the session's state because you could as well use the information that a user is online as a guidance, even if the state would be stored somewhere out of the session. Wouldn't make a difference I guess and storing it in the session looks natural.

The state itself is not in the session. The session is the guide that the user is online. The session contains a listener that is attached to the User. The only real state that resides in the session is the state necessary to batch up any messages that the User has forwarded to the listener in between the HTTP polling requests. If there is an HTML front end, state about that front end will reside in the session as well, but
that's a different issue.

Question: I don't understand why we would need to store all entries in a cache, instead of only keeping the last n entries for each user based on some heuristics such as the last 3 days or something. I would somehow expect that the probability that a user wants to see a message is exponentially decreasing with the messages age. For example that someone wants to see a message that is the 1000 newest message in his timeline is probably almost zero.
Some people mine their timelines for information. I agree that some aging policy is necessary as 36B entries will consume a lot of storage in RAM or on disk, but the last 1,000 is likely too few based on what I have seen of actual user behavior.
In terms of an aging policy in an RDBMS, the cost of aging out old entries is likely to be an index scan or something on that order (DELETE FROM mailbox WHERE date < xxx or a user-by-user DELETE WHERE id IN (SELECT messages > 1000 in mailbox))

Actions

How do I add a content type to a HTTP Post action?

```
http://localhost:1234
header:Content-Type=text%2fxml
<text>%s</text>
```

Platform Specific

Apple

Are there are tips for building on Apples?

Usually, JDK 6 is the best choice. JDK 5 currently doesn't work

---

Table of Contents