

Prototype Implementation Progress

Group:

Hoang Huu **Hanh**

Nguyen Huu **Tinh**

17/05/2004

Content

- Results
- Code fragments of the implemented module
- Demo
- Next Step

Previous Meeting

- Developing module for monitoring and selective parsing the new messages into RDF model.
- Building user interface for the system.
- Creating Inference layer for this prototype supporting for semantic querying.

Results

- Accessing to mailing systems (POP3/IMAP4), periodic checking for **new** messages and retrieving them from.
- Parsing the **new** email messages to RDF model.
- Uniting the new model to existing model persisted in a MySQL database.
 - Accessing and reading the RDF model stored in a MySQL database.
 - Making union of two models.
 - Persisting the union model to a MySQL database

Code Fragments (1/4)

- Accessing mailing system, monitoring for new messages and parsing them to RDF data.

```
Store store = session.getStore("imap");
store.connect(host, username, password);
Folder folder = store.getFolder(INBOX);
folder.open(Folder.READ_WRITE);

// Add messageCountListener to listen for new messages
folder.addMessageCountListener(new MessageCountAdapter() {
    public void messagesAdded(MessageCountEvent ev) {
        Message[] msgs = ev.getMessages();
        System.out.println("Got " + msgs.length + " new
messages");
        for (int i = 0; i < msgs.length; i++) {
            try {
                <... Codes of scattering messages' info to arrays ...>
            }
        }
    }
});
```

Code Fragments (2/4)

- Creating a new model from info stored in arrays

```
Model m = createMM(msgId, from, nfrom, to, nto, subj,  
                  mimeType, sdate, count);  
// dump the new model out  
m.write(System.out); System.out.print("\n");
```

- Merging the new model into existing model stored in a database

```
loadRDB("mailrdf", "root", "", "MailMessages", m);
```

- Cheking for new messages in every 200.000 ms

```
int freq = Integer.parseInt(200000);  
for (; ; ) {  
    Thread.sleep(freq); // sleep for freq milliseconds  
  
    // This is to force the IMAP server to send us  
    // EXISTS notifications.  
    folder.getMessageCount();  
}
```

Code Fragments (3/4)

- Creating a new model from info stored in arrays

Function createMM()

```
Model model = ModelFactory.createDefaultModel();
Property _sn = model.createProperty(SMMURI, sSName);
Property _fr = model.createProperty(SMMURI, sFrom);
Property _rn = model.createProperty(SMMURI, sRName);
Property _to = model.createProperty(SMMURI, sTo);
Property _sb = model.createProperty(SMMURI, sSubj);
Property _mime = model.createProperty(SMMURI, sMime);
try {
    for (int i=0; i < msgcount; i++){
        // Create a new resource for each message with URI is msgID
        Resource msg = model.createResource(msgId[i]);

        // add values of properties of a message
        msg.addProperty(_fr, from[i])
            .addProperty(_sn, ((nfrom[i]!=null)?nfrom[i]:"no-name"))
            .addProperty(_to, to[i])
            .addProperty(_rn, ((nto[i]!=null)?nto[i]:"no-name"))
            .addProperty(_sb, ((subj[i]!=null)?subj[i]:"no-subject"))
            .addProperty(DC.date, sdate[i])
            .addProperty(_mime, mimeType[i]);
    }
}
```

Code Fragments (4/4)

- Uniting the new model to existing model in a database
Function loadRDB()

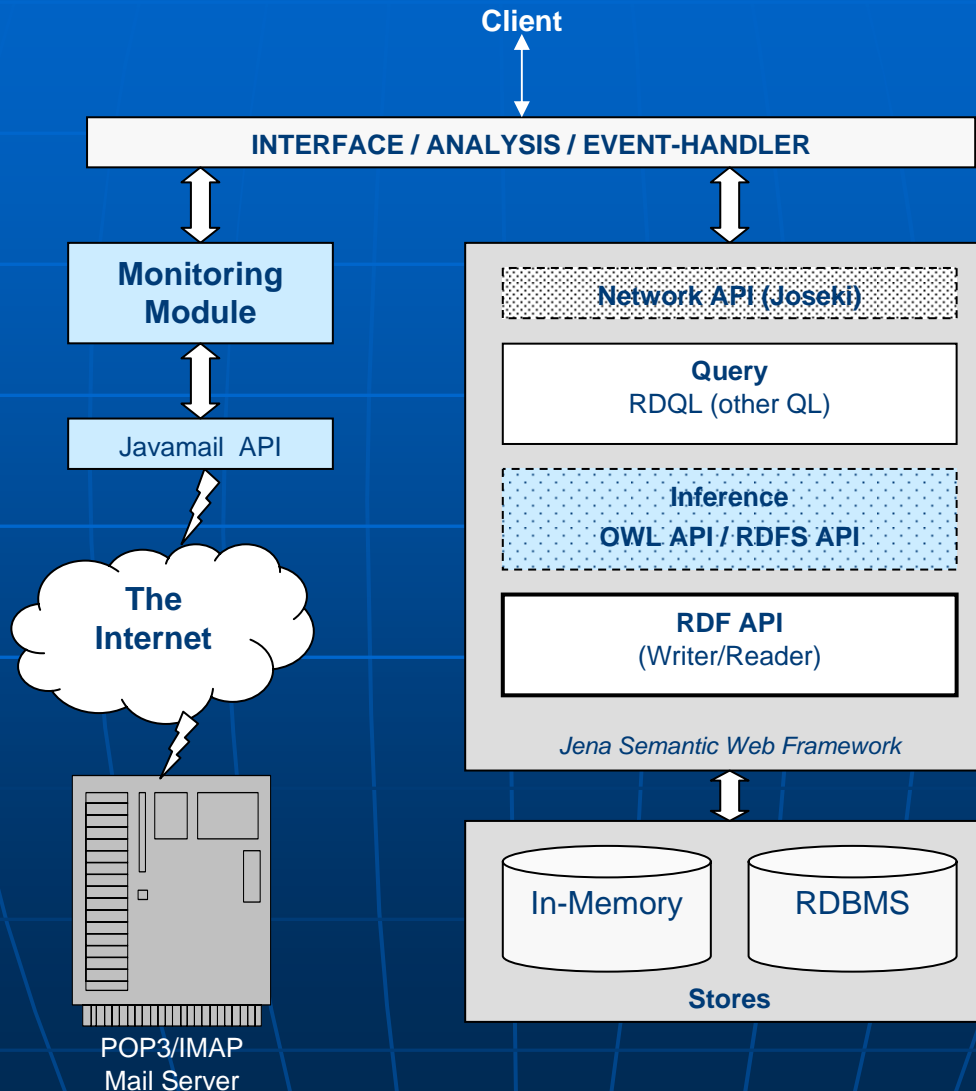
```
// Read a model from the database
ModelRDB em = ModelRDB.open(conn, dbmodel);

// uniting two models and write out a RDF file
Model agg_model = em.union(newmodel);
agg_model.write(new FileOutputStream("messages.rdf"),
                "RDF/XML-ABBREV");

// remove the old model and persist the aggregated
  model into a database
em.remove();
em = ModelRDB.createModel(conn, dbmodel);
em.read(new FileInputStream("messages.rdf"),
        "RDF/XML-ABBREV");
```

Demo...

Prototype Architecture



Next Step

- Creating Inference layer for this prototype supporting for semantic querying.
- Building user interface for the system.

Thank you !