Western Kentucky University
TopSCHOLAR®

Masters Theses & Specialist Projects | Graduate School

1-1-2004

Excel Sheet Based Semantic Email

Rajesekhar R. Dandolu

Follow this and additional works at: http://digitalcommons.wku.edu/theses

Part of the Computer Sciences Commons

Recommended Citation
Dandolu, Rajesekhar R., "Excel Sheet Based Semantic Email" (2004). *Masters Theses & Specialist Projects.* Paper 1101.
http://digitalcommons.wku.edu/theses/1101

This Thesis is brought to you for free and open access by TopSCHOLAR®. It has been accepted for inclusion in Masters Theses & Specialist Projects by an authorized administrator of TopSCHOLAR®. For more information, please contact connie.foster@wku.edu.

EXCEL SHEET BASED SEMANTIC EMAIL

A Thesis
Presented to
The Faculty of the Department of Computer Science
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Computer Science

By
Rajesekhar R Dandolu

December 2004.

EXCEL SHEET BASED SEMANTIC EMAIL

Date Recommended 12/10/2004

Dr. Guangming Xing, Director of Thesis

Dr. Uta Ziegler

Dr. Andrew Ernest

Elmer Gray, Dean of Graduate Studies and Research. 12/16/2004

ACKNOWLEDGEMENTS

I would like to thank my director of thesis Dr. Guangming Xing, for the confidence he showed in me and for his encouragement without which I would not have pursued a master's degree in Computer Science. Dr. Xing's help was also invaluable in researching and producing this document.

I wish to express my warm gratitude to Dr. Uta Ziegler, head of Computer Science Department, for her supportive attitude for my research work.

I would like to thank Dr. Andrew Ernest, Director of Water Resource Center, Western Kentucky University for supporting and funding this thesis work.

I would like to thank my parents for their support and encouragement and my friends, who gave moral support during the completion of the project.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW 8

1.1 INTRODUCTION 8

1.2 BACKGROUND.....8

CHAPTER 2

SEMANTIC EMAIL PROCESS 11

2.1 EXCEL SHEET BASED SEMANTIC EMAIL ARCHITECTURE.....15

2.2 SEMANTIC EMAIL COMPOSER 16

2.3 SEMANTIC EMAIL READER.....20

CHAPTER 3

SEMANTIC EMAIL COMPOSER.....22

3.1 FILE MENU.....23

3.2 SCHEMA MENU 27

3.3 STEPS IN A COMPOSER 35

CHAPTER 4

SEMANTIC EMAIL READER 36

CHAPTER 5

ENVIRONMENT 39

CHAPTER 6

CONCLUSIONS.....40

APPENDIX A.....43

POI-HSSF - JAVA API TO ACCESS MICROSOFT EXCEL FORMAT FILES.....43

NEW WORKBOOK 43

NEW SHEET.....43

CREATING CELLS.....44

READING AND REWRITING WORKBOOKS 44

APPENDIX B 46

JAVAMAIL API.....46

LIST OF FIGURES

Figure 1 Semantic Email Process 11

Figure 2 Excel Sheet Based Semantic Email Architecture 15

Figure 3 Excel Sheet Specifying The Email Format 22

Figure 4 File Menu 23

Figure 5 "New Schema" Sample Outputs 24

Figure 6 Schema Menu 27

Figure 7 "Creating Database" Sample Dialog Box. 28

Figure 8 "Adding Actions" Sample Output. 29

Figure 9 "Inserting Records into Database" Action Details Sample Outputs. 30

Figure 10 "Save Files", Action Details Sample Outputs 31

Figure 11 "Send emails to clients" Sample Outputs 32

Figure 12 "Allowing Duplicate Mails" Schema Setting Sample Dialog Box. 33

Figure 13 "Adding Constraints" Sample Output. 33

Figure 14 “Semantic Email Reader” Sample Frame 38

EXCEL SHEET BASED SEMANTIC EMAIL

Rajesekhar R, Dandolu December 10, 2004. 47 Pages

Directed by: Dr. Guangming Xing

Department of Computer Science Western Kentucky University

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The Semantic Web envisions a portion of the World-Wide Web in which the underlying data is machine understandable and can thus be exploited for improved querying, aggregation, and interaction.

Excel Sheet Based Semantic Email is a type of Semantic Web application, which deals with the understanding of emails received and performing corresponding actions according to the schema specified in the email. The user can compose an email structure and specify all the semantic actions and necessary information related to a particular schema. The emails received are processed according to the schema format to which they belong and corresponding semantic actions are taken.

In this project, Semantic Email is implemented by encoding the information in Excel Sheets. It could be reengineered to support heterogeneous semantic actions based on the particular application.

The project can be enhanced providing a web interface, apart from the email system that is currently used as the way of communication. The clients can directly use the web page, corresponding to the schema rather than sending an email.

Chapter 1

Introduction and Literature Review

1.1 Introduction

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The Semantic Web envisions a portion of the World-Wide Web in which the underlying data is machine understandable and can thus be exploited for improved querying, aggregation, and interaction. Semantic Email is a type of semantic web application, in which the information is exchanged through emails. The semantic information is extracted from the emails, and corresponding actions are taken based on the specification of the Semantic Email application.

1.2 Background

The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. Semantic Web is a key component to the whole theme of e-life on the web.

The Semantic Web is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications. For the Web to reach its full potential, the information should be properly tagged and have well defined meaning. This would create a universally accessible platform that allows data to be shared and processed by automated tools as well as by people [1].

The Semantic Web will provide an infrastructure that enables not just web pages, but databases, services, programs, sensors, personal devices, and even household appliances to both consume and produce data on the web. Software agents can use this information to search, filter,

and prepare information in new and exciting ways to assist the web user. New languages, making significantly more of the information on the web machine-readable, power this vision and will enable the development of a new generation of technologies and toolkits [3].

A typical scenario of the usage of a semantic web would be a semantic search engine. A semantic analyzer analyzes the search data and generates an XML file. The XML file generated specifies the parent-child relationships between various search elements provided. Using the target XML file that was generated, the web is searched semantically. The target XML file is converted into an Xquery by a query generator. Then the semantic web engine takes over searching for path expression [7] matching in its database. Once a match is found the data is retrieved and sent to the user. Giving the user the power to change the search criteria and to refine or change the search data could further extend the search. If the path expression information of the newly refined or changed search data is a sub tree of the former search data then the same resulted data domain applies to the new search data. For example, based on this implementation, a semantic search engine which will search for “country singer Tim McGraw” will result in giving the user all the pages which were the result of a semantic search rather than the result of a brute force text based search which consists of systematically enumerating every possible solution of a problem until a solution is found, or all possible solutions have been exhausted.

The main power of Semantic Web languages is that anyone can create one, simply by publishing some Resource Description Framework (RDF) that describes a set of Uniform Resource Identifiers (URIs), what they do, and how they should be used. RDF Schema and DAML (The DARPA [The Defense Advanced Research Projects Agency] Agent Markup Language) are very powerful languages for creating Semantic Web languages.

Semantic Web applications directly access the logical relationships in the Semantic Web database. Semantic Web applications can efficiently and accurately search, summarize, analyze, and retrieve discrete concepts or entire documents from huge databases.

Semantic Web applications handle both structured and unstructured data. Structured data is stored in relational databases with static classification systems, and also in discrete documents. These databases and documents can be processed and converted to Semantic Web databases and then processed with unstructured data.

Semantic Web database [8] architecture is dynamic and automated. Each new document, which is analyzed, extracted, and stored in the Semantic Web, expands the logical relationships in all earlier documents. These expanding logical relationships increase the understanding of content and context in each document and in the entire database. The Semantic Web conversion process is automated. No human action is required for maintaining a taxonomy, Meta data tagging, or classification. The semantic database is constantly updated and is more accurate.

Semantic Web applications support both human and machine intelligence systems. Humans can use Semantic Web applications on a manual basis and improve the efficiency of search, summary, analysis, and reporting tasks. Machines can also use Semantic Web applications to perform tasks that humans cannot do because of the cost, speed, accuracy, complexity, and scale of the tasks. Examples of Semantic Web applications include military examples like military equipment description ontology and markup.

Chapter 2

Semantic Email Process

The following architecture of a Semantic Email Process is proposed in [3]:

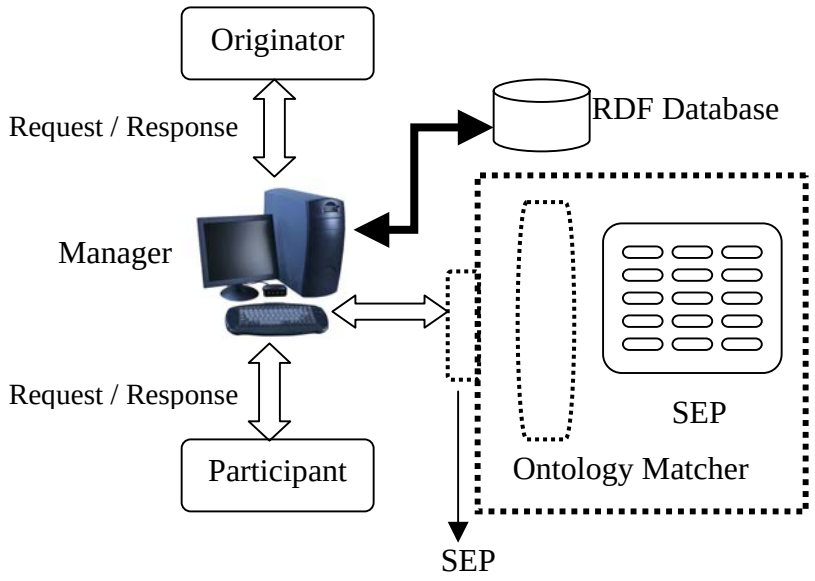

Figure 1 Semantic Email Process

The life cycle of a Semantic Email Process starts with the creation of a generic Semantic Email Process (SEP). The parties that are involved in this process are discussed as follows.

Originator: the originator, who is typically a person, but could be an automated program or agent, initiates a SEP [3]. Here the originator uses a closed authentication system while accepting the responses from the participants.

Manager: The originator invokes a new SEP by sending a message to the Semantic Email manager. The manager sends email messages to the participants, handles responses, and requests changes as necessary to meet the originator's goals. The manager stores all data related to the process in an RDF supporting data set, which may be configured to allow queries by external services (or other managers). To accomplish its tasks, the manager may also utilize external

services such as inference engines, ontology matchers, and other Semantic Web applications, as described further below. The manager may be a shared server or a program run directly by the originator [3]. The manager implements a mechanism similar to the certificate authentication system to identify the responses based on the certificates that are being issued for that SEP by the originator. Every SEP has a unique SEP descriptor called schema name. This schema name is the key for identifying the entities of various SEPs. The schema name is a protected field and cannot be changed by a malicious user. This prohibits the malicious user from responding to an unauthenticated SEP. If malicious user successfully changes the schema name and responds with a valid response entity then the manager rejects the response because the user's name is not being listed in the closed authentication system being implemented.

Participants: The participants respond to messages received about the process. A participant may be a person, a standalone program (e.g., to represent a resource such as a room), or a software agent that acts on behalf of a person (e.g., to respond automatically to requests when possible, deferring others to the person) [3]. The assumption is that the email addresses are used uniquely to determine individuals or sets of potential participants in the process [3].

Interpreting responses: Typically, the originator will provide the participants with a finite set of expected responses. However, suitable reasoning could enable substantially more flexibility. For instance, the originator may initiate an assignment request for a final submission. Then, the manager could use a combination of information extraction or wrapper techniques and/or ontology matching algorithms to map the participant's response into the assignment ontology. There are several interesting outcomes to this mapping. The response may not map to any known element. In this case, the manager may either reject the response or notify the

originator. If the schema name matches but the user is not in the expected list of authorized user, the manager detects an attempt of intrusion and appropriate steps are taken.

Existing email clients provide some features of Semantic Email. Examples of Semantic Email processes are the meeting request feature in Outlook, invitation management via *Evite*, and contact management via *Good Contacts*, which make the Semantic Email process popular. Each of these commercial applications is limited in its scope. Workflow and collaboration systems such as Lotus Notes/Domino and Zaplets offer scripting capabilities and some graphical tools that could be used to implement sophisticated email processes. To illustrate the Semantic Email process, consider several examples:

- **Assignment Submission:**

  Students can submit the assignments to the professor by giving an email to the professor. The interesting thing is that the Semantic Email will read the email sent by the student, grade the assignment, email the student with the grade obtained, and insert the grades into the database.

- **Event Planning:**

  Semantic Email can be used to plan an event like a committee meetings. The Semantic Email Composer sends emails to the committee members informing them about the meeting. The Semantic Email Composer can understand the responses from the committee members and make a decision on the meeting.

- **Course Registration:**

  When the course registration for the students becomes tedious, Semantic Email can be used to solve the problem. The students will be asked to email the courses they need

and the constraints that apply in the course registration. The Semantic Email will respond to the emails from the students and register courses to the students.

- **Small Water Systems:**

  The water resource department needs to send a report about the water quality to the Kentucky Water diversity state agency. The current primary mode of communication being used is the normal postal mailing system. It takes nearly several months for the state agency to collect all the reports from different water resource department, analyze them and make a decision about the water quality. Clearly this is not a good solution. The Semantic Email project can be used in such a scenario by the state agency to collect reports and analyze them in real time. Since, the mode of communication in the Semantic Email is email, it takes considerably less amount of time to collect and analyze the reports.

The Excel Sheet Based Semantic Email project is a two-stage process. The first stage is the Semantic Email Composer, which is used to compose a Semantic Email schema. The schema represents the format of the email, the actions to be performed, and the necessary settings for the actions as well as the schema. The second is the Semantic Email Reader, which reads the contents of the emails received. The Semantic Email Reader finds the hidden schema name based on the contents of the email. The schema name extracted from the email is used to perform the corresponding actions related to the schema.

2.1 Excel Sheet Based Semantic Email Architecture

Though a proposed architecture of the Semantic Email exists, the complete implementation of the architecture is not yet done. The main essence of this project is to implement the proposed architecture of the Semantic Email, named as Excel Sheet Based Semantic Email. Excel Sheets are used as information carriers in this project. This project implements a generic version of Semantic Email. Since, Excel Sheets are widely used by all organizations, using them made the project generic.

The architecture of the Excel Sheet Based Semantic Email is shown in the Figure 2.

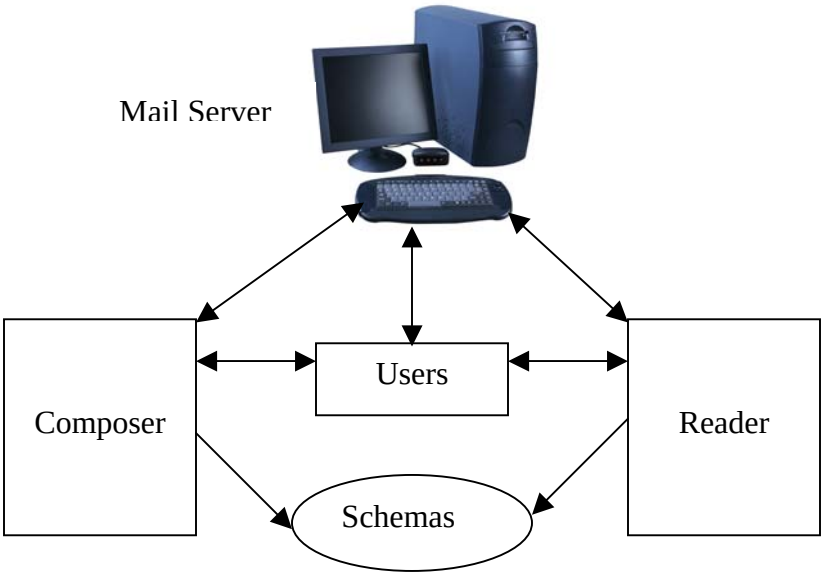

Figure 2 Excel Sheet Based Semantic Email Architecture

Mail server is a normal mailing system that is used to compose and receive emails. Users can directly interact with the mail server using a mail client to send and receive emails. But the interaction between the user and Mail server is not related to the Excel Sheet Based Semantic Email. In order to use the Excel Sheet Based Semantic Email features, users contact either the Composer or the Reader. Users interact with the Composer to define a schema for a particular

application. Composer creates a schema as specified by the user and includes with the list of schemas. Users use the Composer to email clients about the specifications of schema. The Composer must interact with the Mail server to send emails to the clients specified by the user. Users interact with the Reader to perform the semantic actions based on the emails received. Upon the user's request, the Reader reads the emails, processes them, and performs the corresponding semantic actions. The Reader needs to interact with the Mail server to read the emails. The Reader interacts with the list of schemas after learning the schema name from the contents of the received emails.

2.2 Semantic Email Composer

The Email Composer shows an Excel Sheet used for specifying the email format and an extensive set of menu options for defining the schema. Different schemas are defined for different applications and each schema is given a unique name. Defining a schema includes the following:

- Format of the Email
- The actions to be performed
- Database Creation if necessary
- Necessary Schema Settings

For example, consider the schema of inserting student records into the student database. In the above example, the Excel Sheet attached specifies the format of the email. The Excel Sheet holds different values corresponding to different fields in the database. The format of the email must also specify the location of the values in the Excel Sheet. Consider that the student database has three fields: Name, Roll Number, and City. The format of the email specifies the cells in the

Excel Sheet to store the values of Name, Roll Number, and City. The cells in the Excel Sheet are initially filled with the field names indicating to clients the locations where values must be entered. The format of the email is stored in an Excel Sheet format.

The only action to be performed in the above example is to insert a row into the student database. The values of the row to be inserted are obtained from the Excel Sheet received with the email. While specifying the action to be performed, the requirements for that action must also be specified. The action “inserting into database” is specified by telling the location of the cells in the Excel Sheet that contain the values of the row to be inserted. Its row and column number indicates the location of a cell in the Excel Sheet. So, for each field Name, Roll Number, and City in the student database, a cell location in the Excel Sheet must be specified where the value of the corresponding field resides. Some fields in the database may have default values that are not obtained from the Excel Sheet. For each field in the database, there will be a choice to choose between the default value and the value from the Excel Sheet. If the value of a field is the default value, then the default value corresponding to the field should simply be specified. If the value of a field is obtained from the Excel Sheet, then the cell corresponding to the field can be selected from the Excel Sheet displayed. The row and column numbers of the selected cell are automatically stored in the schema.

A database is created only when it is necessary and primarily depends on the action to be performed. For the action “Inserting into Database”, a database is necessary and it is created. Specifying the table name and the table details creates a database. The table details include the table name and data type for each field in the table. For the above example, table name and table details are as follows:

Table Name: Student

Table Details:	Field Name	Field Data Type
	Name	String
	Roll Number	Integer
	City	String

Some of the schema settings are common to all actions and can be specified for all action types. For example, consider the schema setting of allowing duplicate mails. This is a general setting and can be specified with all action types. The schema setting, “allowing duplicate mails,” permits the client to send duplicate mails corresponding to the same schema. By default, any client can send only one mail corresponding to one schema. This default condition can be overridden by setting the “allowing duplicate mails” schema setting to a true value and thus allowing duplicate mails from a client for the same schema.

Most of the schema settings however depend on the action type and vary with the type of action specified. Some of the schema settings that depend on an action type are specified while specifying the action itself. For example, consider the schema setting “Allowing multiple lines,” which is related to the semantic action “Inserting into Database.” The setting “allowing multiple lines” is used to insert multiple rows into the database. The above setting is specified before specifying the action details. Actually, the action details depend on the condition, whether the schema will allow multiple rows or not. If the schema doesn’t allow multiple rows, then the action details must contain the cell details (row and column numbers) for every field in the database that needs values from the Excel Sheet. If the schema does allow multiple rows, then the action details must contain only the column number for every field in the database that needs values from the Excel Sheet.

The schema settings like “allow multiple rows” must be specified along with the action itself. There exist some other schema settings that can be specified explicitly even though they belong to a particular semantic action. Explicitly specified schema settings are also optional. That means that the settings do not need to always be specified along with the schema definition. An illustration of such a type of schema setting is constraints, which are explicitly specified and are optional. For example, consider the case of applying constraints while inserting rows into a database. These constraints are optional, and if the constraints are not specified then the rows are inserted into the database without any constraints. If the constraints are defined, then the rows will be inserted into the database only after satisfying the constraints. The constraints can be of different types; they can be on a single row or on the complete table. Constraints on a single row are restrictions on the values of various fields. For example, consider that the student database and “constraints on a single row” can be as follows:

Name contains “john”

Roll Number in between 630 and 650

Roll Number Less Than 500

City ends with “Green”

Constraints on the complete table will be on aggregate functions like Sum, Average, and Count. If the semantic actions to be performed do not satisfy the constraints for the database, then these actions are ignored. These aggregate functions are applied to the numeric fields in the database. For example, consider an Account database with the Account_No and Balance as fields. Constraints on the Account database can be as follows:

Sum (Balance) <= 60000

Average (Balance) >= 1000

2.3 Semantic Email Reader

The Semantic Email Reader is similar to the email client program used to access the contents of the emails received. However, the Semantic Email Reader is a sophisticated tool that reads the contents of the email received, understands the content, and performs necessary actions based on the content read. The Semantic Email Reader first searches the content for the schema name.

The Semantic Email Reader first gets all the information related to the schema settings. Some of the emails received are rejected based on the schema settings. For example, consider the case of not allowing duplicate mails from a client for the same schema. The duplicate emails received for a particular schema are rejected.

The Semantic Email Reader then searches for the different actions associated with that particular schema. Based on the action, the Semantic Email Reader then searches for the remaining settings and performs the actions according to it. For example, consider the action of inserting records or rows into a database.

For the current example, consider the student database with Name, Roll Number, and City as fields. The values for various fields are obtained either by accessing the corresponding cells in the Excel Sheet or obtaining the default values. Once a complete row is built with all the necessary field values, the Reader connects to the database and inserts the row into the database.

It is necessary to remember that the row is inserted only after satisfying all the constraints that apply to the schema. So, before inserting a row into the database, the Semantic Email Reader must search for the constraints that apply to the schema. Adding the extra row must not violate that behavior of the table.

Some schemas require the status of the action to be replied back to the sender. For example, consider the schema used to save the files attached with the email. The client needs to know the status of the email sent. The schema only saves the files whose file names match with the names in the schema. The schema sends a reply to the client specifying the list of files accepted or saved, list of rejected or unsaved files, and list of files that are not sent and required. The Semantic Email Reader works differently for different action types depending on the action type and schema settings associated with the action.

Chapter 3

Semantic Email Composer

The Semantic Email Composer is a Graphical User Interface (GUI), which displays an Excel Sheet and a set of menu items to define schemas. The Excel Sheet displayed is used to specify the format of the email. All the emails received must have the Excel Sheet attached representing the pattern or format accepted by the Semantic Email Reader. The Excel Sheet contains the schema name that is hidden and locked. The schema name can't be changed, because the cell containing the schema name is locked.

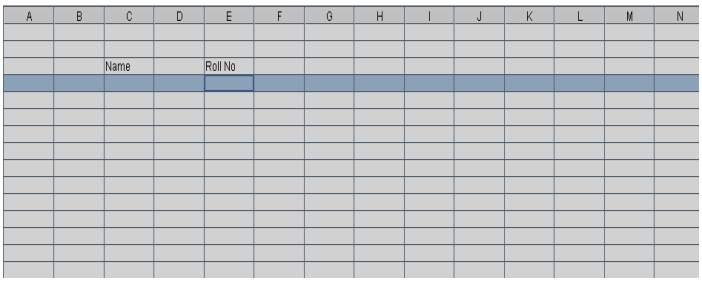

Figure 3 Excel Sheet Specifying The Email Format

The Excel Sheet isn't used in some schemas for specifying the format they accept. But even then, the Excel Sheet must be attached with the email, since the Excel Sheet contains the schema name, which is a required attribute for the Semantic Email Reader. Using the Excel Sheet for representing the format of the email depends on the schema. For example, consider an application that collects student information: for inserting student records into the student database. The locations that are used to store the student information are specified in the Excel Sheet. Instructions about using the Excel Sheet will be given to the clients. A separate Excel

Sheet is specified for each schema defined. The Excel Sheet is displayed on the screen on one of the following two situations:

- When a New schema is selected.
- When a schema that already exists is opened.

The Semantic Email Composer has a set of menus used to define and save schemas. The set of menus in the Composer is mainly divided into two parts

- File Menu
- Schema Menu

The File menu is mainly used to open a schema (new or existing), save the schema after making necessary changes, and quit the Composer. The Schema Menu is used to specify the actions associated with the schema and different schema settings related to the schema.

3.1 File Menu

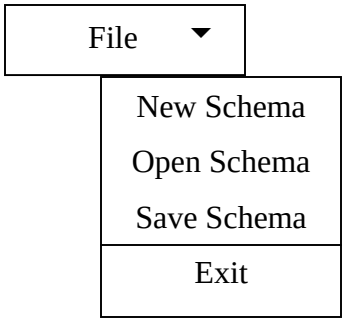

Figure 4 File Menu

The file menu consists of menu items related to opening, saving, and closing a schema. The new schema menu prompts for the schema name, and the schema name must be unique within the system. The Semantic Email Composer checks for the uniqueness of the schema name, and if the schema name already exists, an error message is displayed. A directory with the schema name is

created for each schema. The Composer checks for the existence of the directory in order to check the uniqueness of the schema name. Only three attempts are permitted for giving the schema name. If the number of attempts is exceeded, then the Composer prints an error message, saying that the number of attempts is exceeded.

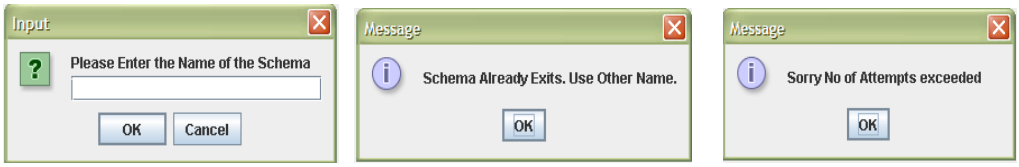

Figure 5 "New Schema" Sample Outputs

Once the valid schema name is given, a directory with the name of the schema is created and an empty Excel Sheet is displayed. The new schema resets all the schema settings to the default values. The Excel Sheet displayed is also an empty sheet.

The open schema menu prompts for a schema name. The schema name given must exist. Checking the existence of a directory with the schema name checks the existence of the schema. The Semantic Email Composer opens a directory with the schema name if and only if a schema exists with that name. If the directory doesn't exist, then it assumes an error and again prompts for a schema name. The Semantic Email Composer gives just three attempts to specify the schema name. Once the number of attempts exceeds the limit, the Composer prints an error message.

The schema information like schema settings and the semantic actions are extracted from the directory. The directory contains an Excel Sheet representing the format of the mail. An xml schema file "actions.xsd" in the directory represents the actions to be performed. An xml schema file "database.xsd" in the directory represents the database information related to the schema. A settings file "settings.inf" in the directory contains all the remaining necessary conditions and

setting related to the schema. The current schema settings are changed to the scheme settings read from the directory. The current semantic actions are also changed to the actions that are read from the directory.

“Save schema” will save all the information including the schema settings, and schema actions related to the current schema into the directory with current schema name. It stores the email format into an Excel Sheet and saves into the directory with the current schema name. The Excel Sheet displayed in GUI is actually a swing component JTable. It is easy to get the information from the JTable. The POI API of the Jakarta project is used to access Excel sheets. The information retrieved from the JTable is stored into the Excel Sheet using the POI API. The Excel Sheet is then saved into the directory with the current schema name.

If a database is associated with the current schema, then the database details must also be stored into the directory with the current schema name. The Semantic Email Composer uses XML schema format to store the database information. The XML schema file with the name “database.xsd” is created and the database information is stored in the XML schema file. The database information includes table name and table details. The table details include the name and data type for each field or column in the database.

The semantic actions are saved in the XML schema file format. A XML schema file named “actions.xsd” is created in the directory with the current schema name. The data related to the semantic actions is stored in the “actions.xsd” XML schema files. The data to be stored in the XML schema file actually depends on the semantic action type. For example, consider the semantic action to save files that are attached with the email. The data to be stored in the XML schema file for the above semantic action is specified below:

- Action name i.e. Save Files
- Number of files
- File names

Similarly, consider another schema action to insert records into a student database. The data to be stored in the XML schema file for the semantic action “insert rows into database” is specified below-

- Action name i.e. Insert rows
- Allow multiple rows or not
- Table name
- Table details
    - For each field specify the field name and data type.

The “Save schema” stores the schema settings into a file named “settings.inf”. The settings file “settings.inf” contains a settings name and the value associated with it for each schema setting. For example, consider the schema settings for allowing duplicate mails. The setting name is “Allow Duplicate Mails” and the value associated with the setting is either “yes” or “no.” But the value associated with the setting is not always as simple as shown above. For example, consider the constraints using aggregate functions. The setting name is “Aggregate constraints.” The values associated with the settings will be as follows

- Aggregate function like Sum, Average, etc.
- Field or Column on which the above aggregate function is applied.
- Relational equation with a numeric value like <= 1000, > 2000, etc.

Another menu item in the file menu is “Quit” that is used to quit or leave the Composer. Remember to save your schema and mail to the client about the schema before quitting the

Composer. If the Composer quits without mailing the clients, the Composer can open the schema later, to complete the mailing part. Opening a schema without any problems requires the schema to be previously saved in a perfect way.

3.2 Schema Menu

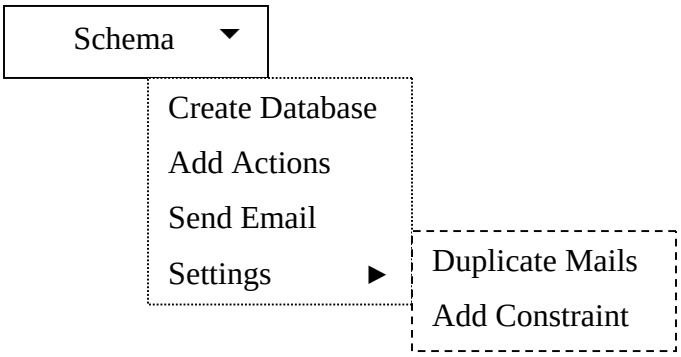

Figure 6 Schema Menu

The schema menu is used to specify the semantic actions, schema settings, and any other schema items like database creation, sending mails to clients, etc.

The menu item “Create Database” is used to create a database related to the current schema. Creating a database is not necessary for all the schemas. Only some schemas with their semantic actions related to database requires database creation. For example, consider the semantic action that inserts student records into the database. A student schema with a semantic action as above requires a database creation. The Semantic Email Composer prompts to enter the database information to create a database. The following represents the database information prompted by the Composer:

- Table Name
- Table Details

For all fields (Columns) in the Table specify

- Field Name
- Field Type

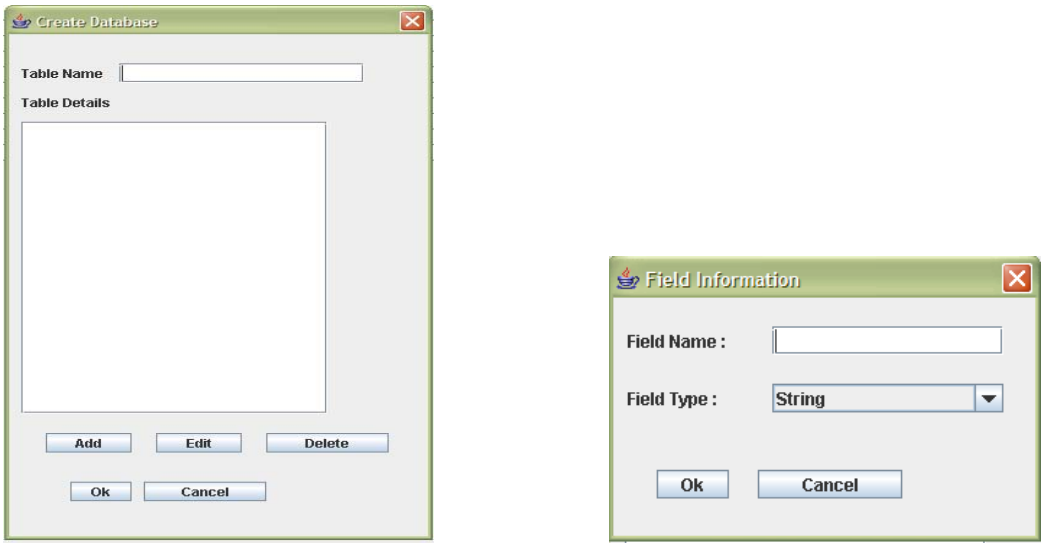

Figure 7 "Creating Database" Sample Dialog Box.

All the above information is necessary to create a database. The student database information that is necessary for the student schema is as follows:

- Table Name: Student
- Table Details:

Field Name	Field Data Type
Name	String
Roll Number	Integer
City	String

Giving all the prompted and required information, as shown above, will create a database.

The menu item “Add Actions” is used to add different semantic actions to the current schema and to specify the required information for that semantic action. The Composer prompts the user to select the semantic action from a list of semantic actions supported. The required

information for an action will depend on the action. To test the base project, two types of actions are implemented.

- Inserting records into database.
- Saving files.

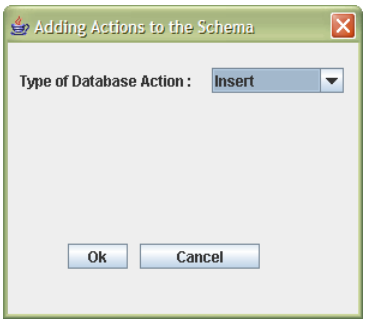

Figure 8 "Adding Actions" Sample Output.

The first action is to insert rows into a database. The required information for inserting a row into a database is as follows:

- Name of the Table to which row must be inserted.
- Schema settings to allow multiple rows.
- Values of the fields (Columns) in the Table.

The first required information “Name of the table” is the name of the table to which the rows are inserted. The schema setting to allow multiple rows is a Boolean value specifying “yes” or “no.” The Boolean value tells whether to insert multiple rows into the database. The value for all the columns in the table is of two types as follows:

- Default value.
- Value extracted from the Excel Sheet.

For each field or column in the table, values are specified based on the type of value. The default values are easily specified by just giving the actual default value for the field. For the values

extracted from the Excel Sheet, the Composer needs the cell location in the Excel Sheet from where the value is extracted. For each field in the table that requires a cell location in the Excel Sheet, specify the row and column in the Excel Sheet to get its value. When multiple rows are to be inserted into the database, specify only the column in the Excel Sheet. When the schema requires multiple rows, all the rows that are not empty on all the fields are selected. The emptiness of a row on a particular field is determined by checking the emptiness of the cell within the row and column related to the field.

Instead of counting the rows and columns for specifying the row and column numbers, the Composer provides a good GUI interface to specify the row and column numbers in the Excel Sheets. The Composer displays a clone for the Excel Sheet. The row and column numbers of a cell can be specified by clicking on that cell in the cloned Excel Sheet displayed. The Composer counts the row and column numbers related to the cell and stores the information.

Figure 9 "Inserting Records into Database" Action Details Sample Outputs.

The second action is “Save files,” which is used to save the files received from the client attached with the email. The action “Save files” prompts the user to specify the number of files and the file names required. File names are required to save only the files required rather than saving all the unnecessary files sent by the client. The Reader will accept only the files specified

in the schema. All the files received will be saved into a directory named “client-name,” where client-name is the name of the client. Clients can send several emails with some of the files to be saved if the “allow duplicate mails” is true.

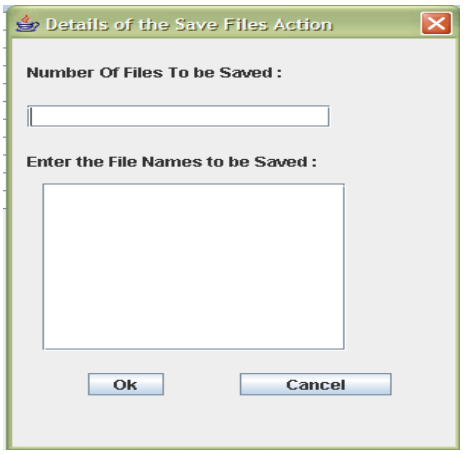

Figure 10 "Save Files", Action Details Sample Outputs

The menu item “Send Email” is used to send emails to the client specifying the format in which the emails must be received. “Send Email” automatically attaches the Excel Sheet to the mail. The Composer just prompts for the email addresses of the clients and the message to be sent along with the email. “Send Email” keeps track of client email addresses into a file “MailList.txt”. This information is used to confirm that the received emails are only from the authorized client. All the remaining mails will be rejected. The mailing list is also used to avoid duplicate mails from a client for a schema.

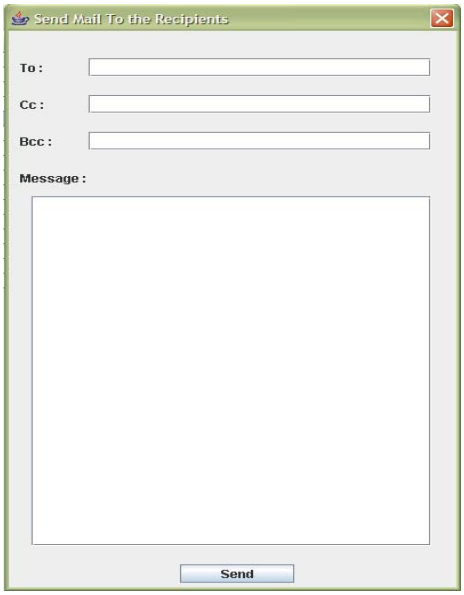

Figure 11 "Send emails to clients" Sample Outputs

The menu item “Settings” is used to specify some of the schema settings related to different actions. On further addition of semantic actions, the options in the “Settings” menu can be extended. Generally, the schema settings related to a particular action are specified while specifying the action itself. The schema settings specified in the “settings” menu are either the common settings to all the different types of actions or optional settings that are not compulsory. The schema settings supporting the actions implemented along with the base project are specified below:

- Allow duplicate emails
- Add constraints.

The first option “Allow duplicate emails” is used to specify whether to allow a client to send multiple mails for the particular schema. By default the Composer sets the schema setting to accept only one email from a client for a particular schema. If the schema explicitly requires multiple mails to be received from a client, this option allows changing the default schema

setting. By making necessary changes to the default schema settings the Composer can allow a client to send multiple emails for a particular schema.

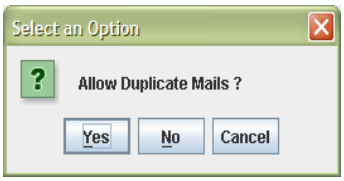

Figure 12 "Allowing Duplicate Mails" Schema Setting Sample Dialog Box.

The next option “Add constraints” is used to specify different constraints on the actions. Adding constraints also depends on the action. The schema setting “Adding constraints” is optional and is related to the semantic action “Inserting into the database.” Constraints on the semantic action “Inserting into a database” can be any of the following:

- Constraints on each row.
- Constraints on Aggregate Functions.

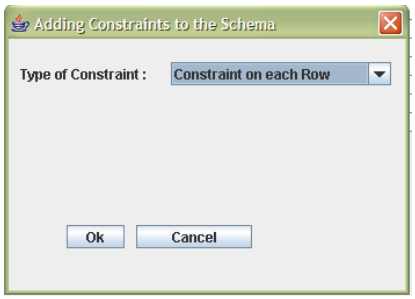

Figure 13 "Adding Constraints" Sample Output.

The first constraint specified above one is a constraint on a single row. The constraint on a single row generally depends on the data type of the field in the table on which the constraint is applied. If the data type of the field is a string, then the constraint will be on how the string value looks. The constraint can use any of the string manipulations described below

- Like operator

  It is similar to the like operator in SQL, which tests for a sub string in the given string value.

  Ex: - Name like “Janes”

- “Starts with” operator

  It checks whether the string value starts with the given sub string.

  Ex: - Name starts with “Mary”

- “Ends with” operator

  It checks whether the string value ends with the given sub string

  Ex: - City ends with “Green”.

If the data type of a field is an integer, then the constraint will be on the value of the field. Constraints on values can be any one of three types described below

- Value less than n
- Value greater than n
- Value in between n1 and n2

Based on the given constraint, check the values obtained from the email, and if they don’t satisfy the conditions, reject the row and the email.

The second type of constraint is on aggregate functions. The following are the aggregate functions checked while inserting rows into the database:

- Sum
- Average
- Count
- Max

- Min

Constraints on aggregate functions are related to the complete table, rather than to the row to be inserted. The constraints will be on how the aggregate function gets affected when the current row to be inserted is inserted into the database. The aggregate functions are chosen from the list above. If the row to be inserted affects the constraint on the aggregate function, then the row is rejected. If the insertion of multiple rows is allowed in the schema, each row is considered individually. A reply email will be sent to the client specifying why the row is rejected.

3.3 Steps in the Composer

The first step is to create a new schema if the required schema doesn't exist. If the schema exists, then select "open schema" menu item by giving the schema name. The format of the email can be changed by using the Excel Sheet displayed. Change the settings for the schema, add necessary actions, and create a database if necessary. After making all the necessary changes, save the schema. The complete schema is now ready. Mail the clients the details of the schema and the required format of the mail.

Chapter 4

Semantic Email Reader

The Semantic Email Reader is similar to the email client program used to access the contents of the emails received. But the Semantic Email Reader is a sophisticated tool that reads the contents of the email received, understands the content, and performs necessary actions based on the content read. The Semantic Email Reader first searches the content for the schema name. The schema name is stored as a hidden value in the Excel Sheet to be sent with the email. So, the semantic mail Reader rejects all the email without an Excel Sheet attachment assuming them to be personal emails or any unauthorized emails. After finding the schema name, the Semantic Email Reader opens the directory associated with the schema.

The Semantic Email Reader first gets all the information related to the schema settings. Some of the emails received are rejected based on the schema settings. For example, consider the case of not allowing duplicate mails from a client for the same schema. Every schema maintains an email list of the clients. The email list is a list of client email addresses that are valid for the current schema. After processing a valid email from a client, the Semantic Email Reader will delete the email address of the client from the email list associated with the schema. Deletion of the email address after processing the email will not allow duplicate mails from a client for the same schema. Even after processing the email, the email address will not be deleted if the schema allows duplicate mails from a client for the same schema.

The Semantic Email Reader then searches for the different actions associated with the schema. Based on the action, the Semantic Email Reader then searches for the remaining settings and performs the actions according to it. For example, consider the action of inserting records or

rows into the database. Before inserting the row, the Reader requires the database details. The database details include the table name and column names associated with their data types.

For the current example, consider the student database with Name, Roll Number and City as fields. Before considering the values for the field, the Semantic Email Reader must verify another schema setting, whether the schema allows multiple row insertion. The Semantic Email Reader then must find whether each field has a default value or the value obtained from Excel Sheet. For each field in the database, if the value is a default value, then find the default value corresponding to the field. If the value of the field is obtained from the Excel Sheet, find a cell in the Excel Sheet where the value is stored. The value of the field is obtained by accessing the cell in the Excel Sheet. Once a complete a row is built with all the necessary field values, the Reader connects to the database and inserts the row into the database.

Remember, the row is inserted only after satisfying all the constraints if the constraints apply to the schema. So, before inserting a row into the database, the Semantic Email Reader must search for the constraints that apply to the schema. Now the Reader checks whether the row to be inserted satisfies all the constraints or not. The constraints can be on a single row or on the complete table. Constraints on the row create a restriction on the values of the row to be inserted. So, the values must satisfy the restrictions to satisfy the constraints. Constraints on the complete table are linked with a behavior of the table. Adding the extra row must not violate that behavior of the table. For example, consider the aggregate functions like Sum, Average, Count, Min, Max, etc, which are constraints on the complete table. Consider the aggregate function Average with a constraint as Average (Balance) > 1000. The aggregate function Average is evaluated on the column “Balance” which means that the Average of the balances in the table must always be greater than 1000. If the values of the row to be inserted changes the average and makes it to go

below 1000, then the row is rejected. Some schemas require the status of the action to be replied back to the client.

For example, consider the schema used to save the files attached with the email. The client needs to know the status of the email sent. The schema compares the file names in the email with the files names associated with the schema. The schema only saves the files whose file names matched with the names in the schema. The schema sends a reply to the client specifying the list of files accepted or saved, list of rejected or unsaved files, and list of files that are not sent and required. The Semantic Email Reader works differently for different action types depending on the action type and schema settings associated with the action.

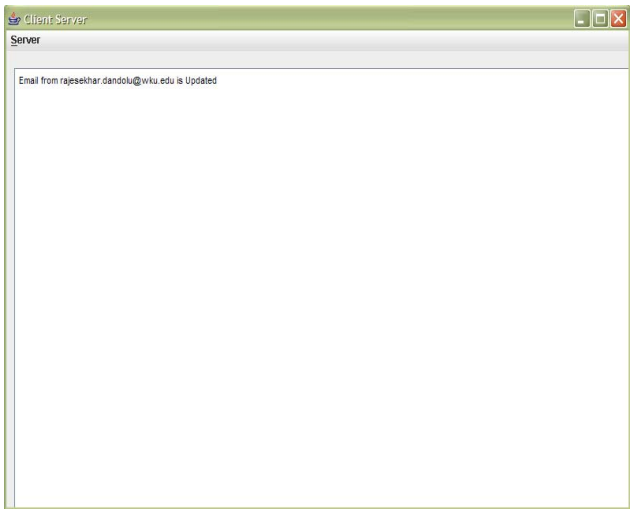

Figure 14 “Semantic Email Reader” Sample Frame.

Chapter 5

Development Environment

The primary environment used to develop the project is JAVA. Since JAVA is machine independent, there is no restriction on the operating system used. The version of the JAVA used is J2SE1.5.0. The basic Application Program Interface (API) of Java Development Kit (JDK) 1.5.0 doesn't provide all the classes required for the project. So, another version of JAVA J2EE 1.4.2 SDK is downloaded and the required classes are imported. The following are the jar files imported from the J2EE 1.4.2 SDK

- mail.jar
- activation.jar
- j2ee.jar
- j2ee-svc.jar

Apart from all these classes, some other classes were downloaded from the Apache Jakarta project. Apache Jakarta has a module called POI, which consists of complete API to access Microsoft format files. Only the jar files from the HSSF part of the POI project are downloaded for dealing with the Excel Sheets in JAVA.

Chapter 6

Conclusions

The Excel Sheet Based Semantic Email is a type of Semantic Web application, used to receive, understand, and perform semantic actions based on the information in the email and the schema that the email is conforming to. The existing Semantic Email systems had many restrictions and are not generic. It is difficult to use in practice as either special software is required or in-depth knowledge of XML is needed. In this project, Excel sheets are used as information carriers since they are widely used. This makes the software more accessible to users without in-depth computer skills. The Excel Sheet Based Semantic Email is implemented supporting two types of semantic actions: file extraction and database operations. The Excel Sheet Based Semantic Email is a base project and numerous applications can be developed from it.

As future work, this project can be further enhanced by implementing the following features:

- A web interface of the Excel Sheet Based Semantic Email can be developed to enhance the project features.
- The project can be extended by adding more semantic action types: applying more constraints, etc.
- The current Excel Sheet Based Semantic Email can be a good starting step for applications where data exchange is extensive.
- In this thesis work, a very primitive way of implementing the schema repository is used. It is interesting to investigate on how to improve the implementation of schema and data repository, so that large-scale applications can be accommodated.

Bibliography

[1] Hendler, James, Berners-Lee, Tim and Miller, Eric "*Integrating Applications on the Semantic Web*," Journal of the Institute of Electrical Engineers of Japan, Vol 122(10), October, 2002, p. 676-680

[2] John Domingue, Martin Dzbor: *Magpie: supporting browsing and navigation on the semantic web*, In Proc of the 9th international conference on Intelligent user interface, p. 191-197, 2004.

[3] Luke McDowell, Oren Etzioni, Alon Halevy, and Henry Levy: *Semantic Email*, In Proc of the 13th international conference on World Wide Web, p. 244-254, 2004.

[4] Bernardo Cuenca Grau: *A possible simplification of the semantic web architecture*, In Proc of the 13th international conference on World Wide Web, p. 704-713, 2004.

[5] Panos Constantpoulos, Vassilis Christophides, Dimitris Plexousakis:*Conference review: Semantic Web Workshop:: models, architectures and management*, intelligence, Vol 12, Issue 2, p. 39-44, 2001.

[6] T.Berners-Lee, J. Hendler, and O.Lassila: *The Semantic Web*, Scientific American, May 2001.

[7] Zografoula Vagena, Vassilis J. Tsotras: *Path-expression Queries over Multi version XML Documents*, June 2003.

[8] Claudio Gutierrezuc, Carlos Hurtadouc, Alberto O. Mendelzonut: *Foundations of Semantic Web Databases*, September 2003.

[9] R.Guha, R.McCool, R.Fikes: *Contexts for the Semantic Web*, 2003.

[10] Aimilia Magkanaraki, Val Tannen, Vassilis Christophides, Dimitris Plexousakis: *Viewing the Semantic Web Through RVL Lenses*, 2002.

[11] Nenad Stojanovic, Rudi Studer, Ljiljana Stojanovic: *An Approach for the Ranking of Query Results in the Semantic Web*, 2002.

[12] Peter F. Patel-Schneider and Jerome Simeon: *Building the SemanticWeb on XML*, 2002.

Appendix A

POI-HSSF - Java API To Access Microsoft Excel Format Files

New Workbook

Microsoft Excel uses the term workbook to represent an Excel file. Creating an Excel file means implies creating a new Workbook using POI. A new workbook can be created using POI as shown below

```
HSSFWorkbook wb = new HSSFWorkbook();
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

New Sheet

A workbook can consist of many Excel Sheets in it. After creating a new workbook, new sheets must be created within the workbook. Excel Sheets are the actual containers of data. New Excel Sheets can be added to the workbook as shown below.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet("new sheet");
HSSFSheet sheet2 = wb.createSheet("second sheet");
```

Creating Cells

Cells within an Excel Sheet must be created explicitly and can be filled with different types of values:

```
// Create a row and put some cells in it. Rows are 0 based.
HSSFRow row = sheet1.createRow((short)0);
// Create a cell and put a value in it.
HSSFCell cell = row.createCell((short)0);
cell.setCellValue(1);
// Or do it on one line.
row.createCell((short)1).setCellValue(1.2);
row.createCell((short)2).setCellValue("This is a string");
row.createCell((short)3).setCellValue(true);
```

Reading and Rewriting Workbooks

Reading the workbook and its contents is similar to writing and creating the workbook.

```
POIFSFileSystem fs  = new POIFSFileSystem(new
                                    FileInputStream("workbook.xls"));
HSSFWorkbook wb = new HSSFWorkbook(fs);
HSSFSheet sheet = wb.getSheetAt(0);
HSSFRow row = sheet.getRow(2);
HSSFCell cell = row.getCell((short)3);
```

```
if (cell == null)
    cell = row.createCell((short)3);
cell.setCellType(HSSFCell.CELL_TYPE_STRING);
cell.setCellValue("a test");

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

Appendix B

JavaMail API

The JavaMail API is a package for reading, composing, and sending electronic messages. The JavaMail API is designed to provide protocol-independent access for sending and receiving messages.

Reviewing the Core Classes

The following Core Classes those are required for sending and receiving messages.

- Session

The Session class defines a basic mail session. The Session object takes advantage of a java.util.Properties object to get information like mail server, username, password, and other information that can be shared across your entire application.

- Message

The Message class is used to define the message to send, after having the Session object. Since, Message is an abstract class, its subclass, MimeMessage is used. A MimeMessage is an email message that understands MIME types and headers, as defined in the different RFCs. Message headers are restricted to US-ASCII characters only, though non-ASCII characters can be encoded in certain header fields. Once having the message, its parts can be set since Message implements the Part interface.

- Address

The message needs to be addressed, after creating it using Message class. Similar to Message class, Address class is also an abstract class. Since Address is an abstract class, its subclass InternetAddress is used. InternetAddress class is used to specify the email address in the To, Cc and Bcc fields of the message.

- Authenticator

The Authenticator class accesses protected resources via a username and password. Since Authenticator is an abstract class, subclass the Authenticator class and return a PasswordAuthentication instance using getPasswordAuthentication() method.

- Transport

Using the Transport class is the final part and it is used to transport or send the message via the network.

- Store & Folder

Using the Session object, a Store object can be created. Store class is used to connect the mailbox and access the mail folders using Folder class.