THREAD NAVIGATION

Background

Electronic communication has become an integral part of an individual's life both at business and at home. Through the use of electronic communications, such as email, Instant Messaging, SMS, and the like, individuals are able to keep in touch, transact business and set up meetings. For example, some personal information manager (PIM) programs allow a user to manage and organize e-mail messages, schedules, tasks, notes, contacts, and other information.

In order to help a user navigate their e-mail messages more easily they may group messages by attributes such as: date, size, conversation, subject, importance, and the like. Many users read their email messages online. It is often hard, however, to read a long email conversation given the length of the contents and the numerous replies aggregated into a single thread.

Summary

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

A user may easily read, browse, and jump through the messages contained within a thread. User interface elements make it easier for a user to identify the messages that are contained within the thread. Different portions of the message may be highlighted such that they are easily identifiable. The user may easily navigate through the thread by selecting a user interface element, such as an arrow button, to move to the next or previous message within the thread. The order of the thread may also be reversed such that the user may view the thread from the first message to the last message, or view the thread from the last message to the first message. The messages within the thread may be marked, such as by using XML tags, in order to facilitate the identification of the sections of the messages within the thread.

Brief Description of the Drawings

FIGURE 1 illustrates an exemplary computing device;

FIGURE 2 is a functional block diagram generally illustrating a thread navigation system;

FIGURE 3 shows an exemplary thread;

FIGURE 4 illustrates a user interface element for navigating messages within a thread;

FIGURE 5 shows reversing the order or the messages within a thread;

FIGURE 6 illustrates tagging threads;

FIGURE 7 shows a complex thread; and

FIGURE 8 shows a process for navigating through a thread, in accordance with aspects of the invention.

Detailed Description

Referring now to the drawings, in which like numerals represent like elements, various aspects of the present invention will be described. In particular, FIGURE 1 and the corresponding discussion are intended to provide a brief, general description of a suitable computing environment in which embodiments of the invention may be implemented.

Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Other computer system configurations may also be used, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. Distributed computing environments may also be used where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

When reading the discussion of the routines presented herein, it should be appreciated that the logical operations of various embodiments are implemented (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations illustrated and making up the embodiments of the described herein are referred to variously as operations, structural devices, acts or modules. These operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof.

As used herein, the term "message" refers to the region of a thread that is from the beginning of one message header within the thread to the beginning of the next message header that is within the thread.

Referring now to FIGURE 1, an illustrative computer architecture for a computer 2 utilized in the various embodiments will be described. The computer architecture shown in FIGURE 1 illustrates a conventional desktop or laptop computer, including a central processing unit 5 ("CPU"), a system memory 7, including a random access memory 9 ("RAM") and a read-only memory ("ROM") 11, and a system bus 12 that couples the memory to the CPU 5. A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM 11. The computer 2 further includes a mass storage device 14 for storing an operating system 16, application programs, and other program modules, which will be described in greater detail below.

The mass storage device 14 is connected to the CPU 5 through a mass storage controller (not shown) connected to the bus 12. The mass storage device 14 and its associated computer-readable media provide non-volatile storage for the computer 2. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, the computer-readable media can be any available media that can be accessed by the computer 2.

By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, digital versatile disks ("DVD"), or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer 2.

According to various embodiments of the invention, the computer 2 may operate in a networked environment using logical connections to remote computers through a network 18, such as the Internet. The computer 2 may connect to the network 18 through a network interface unit 20 connected to the bus 12. The network interface unit 20 may also be utilized to connect to other types of networks and remote computer systems. The computer 2

may also include an input/output controller 22 for receiving and processing input from a number of other devices, including a keyboard, mouse, or electronic stylus (not shown in FIGURE 1). Similarly, an input/output controller 22 may provide output to a display screen, a printer, or other type of output device.

As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 14 and RAM 9 of the computer 2, including an operating system 16 suitable for controlling the operation of a networked personal computer, such as the WINDOWS XP operating system from MICROSOFT CORPORATION of Redmond, Washington. The mass storage device 14 and RAM 9 may also store one or more program modules. In particular, the mass storage device 14 and the RAM 9 may store an email program 24 and a word processing application program 10. The word processing application program 10 is operative to provide functionality for creating and editing electronic documents, such as an email for email program 24. According to one embodiment of the invention, the word processing application program 10 comprises the MICROSOFT WORD word processing application program from MICROSOFT CORPORATION. Other word processing applications may also be utilized.

The email application program 24 is operative to provide functionality relating to emails. For example, email program 24 may be used for creating, displaying, sending and receiving emails. According to one embodiment of the invention, the email application program 24 comprises the MICROSOFT OUTLOOK email application program from MICROSOFT CORPORATION. Other email and/or messaging applications may also be utilized.

In conjunction with the displaying of an email, an email layout manager 26 may be utilized to format the email for easier navigation. Although email layout manager 26 is shown separately from email program 24, this functionality may reside in many places. For instance, it may be included in the email program or at some other location. Generally, email layout manager 26 configures a thread such that it may be easily navigated.

FIGURE 2 illustrates a thread navigation system 200, in accordance with aspects of the invention.

Generally, thread navigation system 200 helps a user to read, browse, and jump through the messages that are contained within a thread. The layout manager 26 provides services for email application 24. According to one embodiment, email documents and corresponding threads may be stored in a backing store 34. In order to facilitate communication with the layout manager 26, one or more callback routines, may be

implemented. Through the use of the callback code, the layout manager 26 may query for additional information necessary to configure various portions of threads.

The layout manager 26 provides facilities for formatting an email and navigating through a thread. The layout manager 26 provides these facilities in response to a request from the email application program 24. Email program 24 and layout manager 26 includes navigation tools to make it easier to read, browse, and jump through relevant portions of an email thread. Layout manager 26 may include user interface elements within an email to make it easier for a user to identify and navigate the messages that are contained within the thread. Layout manager 26 may also highlight different portions of the message such that they are more easily identifiable. When a user interface element is contained within the thread, a user may easily navigate through the thread by selecting a user interface element, such as an arrow button, to move to the next or previous message within the thread. Layout manager 26 may also change the order of the messages within the thread before it is rendered. For example, the order of the thread may be reversed such that the user may view the thread from the first message that was sent in the thread to the last message (most current message), or view the thread from the last message to the first message. Layout manager 26 may also mark the thread, such as by using XML tags, in order to facilitate the identification of the sections of the messages within the thread.

The email program 24 may communicate with the layout manager 26 to request that at least a portion of the thread be formatted. The email program 24 may also provide to the layout manager 26 the text and other content from the email that should be formatted. Tagging portions of an email thread allows email program to more easily identify sections within the thread. For example, XML markup may be used to tag the header portion of an email such that it may easily be identified. Email program 24 is also configured to render an email document 40.

FIGURE 3 shows an exemplary thread. As illustrated, email thread 300 shows five different messages (320, 321, 322, 323 and 324) within the email thread 300. According to one embodiment, the header and/or beginning of each message is shaded (See 310, 311, 312 and 313) so that the user can easily identify where the different messages begin and end. Any way of highlighting the individual messages within the thread may be used. For example, the messages could be shown in one color and the header information could be shown in a different color. According to other embodiments, the headers may be displayed in many different manners. For example, they could be shown collapsed, semi-collapsed or fully expanded. Other information may also be stored in the header. For instance, the header

could include a number (330) indicating the number of the current message within thread. In the present example, the second header 310 indicates that it is the second of five total messages (330).

According to one embodiment, when a user hits reply 340, the header information gets tagged and is included in the reply. One method to identify headers is to use heuristics to determine where that header is, by looking for fields within the thread, such as: "from" "to"; "subject", etc. (See box 350). Another alternative to locating the headers is to tag the headers. For instance, XML markup may be used to indicate where the header is (See Figures and related discussion below). Heuristics can reasonably determine where a header may begin and end, but using XML markup a program may be certain as to who the sender was and who the recipients were and what the subject is and what the date is. According to one embodiment, the headers are gradient shaded (310-313) so the header stands out. This shading helps the user visually parse the thread. A user can quickly scroll down through the email thread and know that when the shading begins the next message within the thread begins.

The header may also be distinguishable based on characteristics such as who sent the email, the dates, or other information. For example, the headers associated with one user may be colored red (not shown) whereas the headers associated with another user may be colored green.

FIGURE 4 illustrates a user interface element for navigating messages within a thread. A user interface element may provide the user with a way to navigate from one message to the next within the thread without having to scroll through the thread. According to one embodiment, a next arrow 410 and a previous arrow 420 allow a user to navigate to the next message or previous message within the thread.

According to one embodiment, when the user hovers over a shaded area of the header 430, the two navigational arrows (410 and 420) appear and when selected the displayed portion of the thread quickly jumps to the next header or to the previous one within the thread. In this way, the user can place their cursor over an arrow and keep clicking to move through the thread without having to move their mouse to select another message. This allows a user to easily navigate the messages within the email thread by going to each message within the thread very quickly. For example, if a user is looking for somebody's specific reply, the user could keep clicking one of the navigational arrows to navigate the e-mail thread until that reply is located.

The use of the navigational arrows is an alternative to scrolling through the email thread to search for something. The user can visually look at one portion of the screen and keep clicking one of the navigational arrows and eventually click to bring the message to the portion of the screen the user is looking. According to one embodiment of the invention, the user may select a user interface element to move to the start or end of the email thread. Going to the bottom of the email allows the user to chronologically read the email. A user can read what the first person said and then either click the up arrow or scroll to go to the next thread.

FIGURE 5 shows reversing the order or the messages within a thread. There are several things the user may want to do when reading a thread. One is to read the thread in chronological order. Without reversing the thread, the user would have to navigate to the bottom of the thread and then read the thread from the bottom to the top.

The messages within a thread may be reversed such that the thread begins with the first sent message within the thread. According to one embodiment, a button called "Start Conversation with Original Email" may be used to reverse the entire thread so that the oldest reply in the entire thread is first. Threads typically start with the most recently sent message and goes down to the oldest, which is the original message within the email thread. Reversing the order allows the user to read the entire thread in order, which is often desirable for long threads.

Referring to the original order 510, three messages are contained within the thread. The first message (530) was sent November 14, 2004 and is displayed last in thread 510. The second message (540) was sent November 15, 2004 at 9:59 PM and is displayed second in thread 510. The third message (550) was sent November 15, 2004 at 11:13 PM and is displayed first in thread 510. While this example thread may be easily seen in its entirety, many threads are much longer and it may be difficult for a user to navigate through the thread.

In reversed thread 520, the messages contained within the thread are reversed such that the user may now see the first message that was sent in the thread before viewing the next message that was sent in the thread. When reversed, message 530 is now displayed first in thread 520, followed by message 540 that is then followed by message 550.

FIGURE 6 illustrates tagging threads. Threads are tagged so that portions of a thread may be easily identified. According to one embodiment, the thread is tagged according to an XML schema. Many different marking schemes may be utilized. According to one embodiment, the markup is concise to conserve space. Referring to sample email 610

it can be seen that Roberto Taboada is the sender; Deborah Harris is the receiver; the subject is "Dinner" and the message was sent Friday, October 22, 2004 at 2:01 PM. The body of the message is "Where do you want to go for dinner? Roberto." According to one embodiment, each item within the message is tagged with an element. The XML Markup 620 of email 610 illustrates exemplary tagged elements. The following table illustrates an exemplary description of the elements:

element	description
<html xmlns:e="urn:schemas-microsoft-com:office:mailheader">	Namespace declaration that goes at the beginning of the email (this may be changed to meet any namespace standards).
e:	Namespace prefix that accompanies all elements, tying them to the new namespace (this may be easily changed so as to not conflict with any existing namespaces).
m	Message
f	From
s	Sent
t	To
c	CC
b	BCC
a	Subject
v	Value: the actual value of the field. According to one embodiment, the From field does not distinguish between individual recipients.

Alternatively, only a portion of the elements within the email may be tagged. For instance, all of the header information could be tagged as a single unit.

FIGURE 7 shows a complex thread 700. The messages within a thread 710 may come from various email programs. For example, a first message may be generated by program 1, whereas the rest of the messages within a thread may be generated by program 2. According to one embodiment, when a thread is received, it is parsed to locate the header information. The parsing identifies the header blocks for the email messages that are authored using email programs that do not utilize a markup. According to one embodiment,

temporary tags are added to the thread before it is rendered. The tags may also persist. The temporary tagging helps provide the navigation features described herein when legacy versions of an email composer (or other email clients) are involved.

The XML Markup that is used to mark the email is concise in order to avoid bloat in the thread. As illustrated, very simple tags are used such that the size of the email is not significantly affected by the XML markup 720. XML markup 720 is the markup of sample email 710.

FIGURE 8 shows a process for navigating through a thread, in accordance with aspects of the invention. After a start operation, the process flows to block 810 where a thread is accessed.

Flowing to operation 820, the thread is parsed to determine the location of items of interest. As discussed above, according to one embodiment, the header for each message is an item of interest. Any item, however, within the thread may be designated as an item of interest. The parsing examines the thread for locations within the thread that are already tagged; identifies the ranges of the thread that were already tagged and then tags the areas of the thread that were not. At this point, the locations of the headers are known.

Moving to operation 830, the appearance of the thread, such as the headers, may be modified. For example, gradient shading could be applied to the headers; the headers could be colored; compressed/expanded; and the like. Generally, any action that is defined may be applied.

At operation 840, the thread is rendered and presented to the user for navigation through the thread. The rendered thread includes the modified appearance such that a user may more easily navigate the thread.

Flowing to operation 850, interactions with the thread are monitored to determine whether the user has activated any of the navigational features as described above. For example, the process may monitor whether the user hovers over a header. The process may also monitor to determine when a user interface button is selected.

Transitioning to operation 860, any determined navigational actions are performed. For example, according to one embodiment, when a user hovers over a header, the next and previous arrows are shown. The action may also be changing the display based upon the action of selecting a user interface control. For example, a control may be selected to reverse the thread, move to the top/bottom of the thread, move to the next/previous message within the thread, and the like.

The process then moves to an end operation and continues processing other actions.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention.

Claims

WHAT IS CLAIMED IS:

1. A computer-implemented method for navigating messages within a thread, comprising:

accessing the thread;

determining a location for each message header within the thread;

modifying an appearance of each message within the thread based at least in part on the location for each message header; and

rendering the thread.

2. The computer-implemented method of Claim 1, wherein modifying the appearance of each message within the thread comprises performing at least one of: changing a shading of each message header; applying a gradient shading to each message header; changing a color of each message header; collapsing at least a portion of the message header; expanding at least a portion of the message headers; including an identifier within each message header; and including a selectable user interface element within at least one of the message headers that when selected performs a navigational function relating to the messages within the thread.

3. The computer-implemented method of Claim 1, wherein determining the location for each message header within the thread comprises applying heuristics to the thread to identify locations of the message headers.

4. The computer-implemented method of Claim 3, wherein determining the location for each message header within the thread comprises locating at least one of: a From field; a To field; a Subject field; and a Message field.

5. The computer-implemented method of Claim 1, further comprising tagging the thread using a markup language; wherein tagging the thread comprises tagging at least one element that is associated with each message.

6. The computer-implemented method of Claim 5, wherein the element is at least one of: a message header element; a message element; a from element; a sent element; a to element; a cc element; a bcc element; a subject element; and a value element.

7. The computer-implemented method of Claim 5, wherein the tagging is temporarily associated with the thread.

8. The computer-implemented method of Claim 5, wherein the markup is an XML markup.

9. The computer-implemented method of Claim 1, further comprising determining when a user is hovering over one of the messages and displaying a navigational control that may be selected to navigate from one message within the thread to another message within the thread.

10. The computer-implemented method of Claim 9, wherein the navigational control is displayed such that it remains in approximately the same location of a screen as the thread is navigated.

11. The computer-implemented method of Claim 1, further comprising reversing the order of the messages within the thread.

12. A computer-readable medium having computer-executable instructions for navigating messages within a thread, comprising:

accessing the thread;
parsing the thread to identify each message header;
tagging each message header according to a schema;
modifying the appearance of each message header; and
rendering the thread.

13. The computer-readable medium of Claim 12, wherein the schema is an XML schema.

14. The computer-readable medium of Claim 13, wherein parsing the thread to identify each message header comprises locating at least one of: a From field; a To field; a Subject field; and a Message field.

15. The computer-readable medium of Claim 12, wherein tagging each message header according to the schema comprises tagging: a from field; a sent field; a to field; a cc field; a bcc field; and a subject field.

16. The computer-readable medium of Claim 12, further comprising displaying a selectable navigational control within the thread and when the selectable navigational control is selected modifying the rendering of the thread.

17. The computer-readable medium of Claim 16, wherein the selectable navigational control performs at least one of: moving to a next message; moving to a previous message; moving to a top of the thread; moving to a bottom of the thread; and reversing the thread.

18. A system for navigating messages within a thread, comprising:

means for accessing the thread;

means for determining a location for each message header within the thread;

means for modifying an appearance of each message header;

means for navigating within the thread using at least one navigational control that is displayed within a message header; and

means for rendering the thread.

19. The system of Claim 18, wherein means for modifying the appearance of each message comprises changing the shading of each message header.

20. The system of Claim 19, further comprising means for tagging the thread according to an XML schema.