Leopold–Franzens–Universität
Innsbruck

Institut für Informatik
Forschungsgruppe DPS
(Distributed and Parallel Systems)

Benutzerunterstützte Emailreduktion Teil 2

Bachelorarbeit

Betreuer: Dr. Barbara Weber
Dr. Michael Welzl

Thomas Nolf (0315328)

Thomas.Nolf@student.uibk.ac.at

Innsbruck
16. Juli 2008

Zusammenfassung

Emails werden oft dazu benutzt zeitlich begrenzt relevante bzw. gültige Information auszutauschen. Durch die Asynchronität des Mediums Email kommt es dabei relativ häufig vor, dass Nachrichten zum Zeitpunkt des Lesens diese begrenzte Dauer bereits überschritten haben und somit für den Leser irrelevant sind. Bis jetzt obliegt es dem Empfänger abgelaufene Emails zu identifizieren und zu löschen, was gerade nach längerer Abwesenheit einen erheblichen Zeitaufwand darstellen kann.
In dieser Arbeit wurde eine Lösung entwickelt, die es erlaubt Nachrichten mit einem Ablaufdatum zu versehen und dadurch zur Reduktion von Emails beitragen soll.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Email (Electronic Mail) ist heute ein weit verbreitetes und oft eingesetztes Kommunikationsmedium. Die Asynchronität dieses Mediums bringt viele Vorteile aber auch Nachteile mit sich. Ein solcher Nachteil ist, dass Nachrichten vom Empfänger häufig zu spät gelesen werden und dadurch der Inhalt bereits nicht mehr aktuell oder ungültig ist. Als Beispiele seien hier verschiedene Einladungen zu Vorträgen, Präsentationen, Meetings und anderen zeitlich festgelegten Veranstaltungen und Aktivitäten genannt, die im Nachhinein für den Empfänger oft nicht mehr relevant sind. Der Empfänger muss diese Emails erst öffnen und teilweise lesen um abgelaufene Nachrichten identifizieren und anschließend löschen zu können.

Ziel dieser Arbeit ist es eine Möglichkeit zu schaffen um Emails mit einem Ablaufdatum versehen zu können. Weiters soll untersucht werden, wie man diese Funktion als Standard einbringen kann, sodass diese künftig von Entwicklern in verschiedene Emailclients eingebunden und die Verwendung dadurch einem breiten Benutzerspektrum ermöglicht wird. Außerdem soll in dieser Arbeit anhand einer Beispielimplementierung gezeigt werden, wie eine solche Lösung aussehen könnte.

In Kapitel 2 wird eine bestehende Lösung analysiert und deren Funktionsweise erläutert. Es folgt ein Kapitel, in dem auf Internetstandards eingegangen wird. Es wird ein Headerfeld definiert welches konform zum Emailstandard ist und untersucht, welche Vorgehensweise notwendig wäre um dieses Headerfeld in den Standard zu bringen. Weiters wird in diesem Kapitel eine Referenzimplementierung vorgestellt, welche im Zuge dieser Arbeit erstellt wurde und eine mögliche Verwendung dieser Funktion beispielhaft darstellen soll. In Kapitel 4 wird beschrieben, wie die Lösung getestet wurde. Abschließend erfolgt in Kapitel 5 eine Zusammenfassung der verwendeten Software und ein Rückblick.

2 Analyse bestehender Ansätze

Die Möglichkeit Emails mit einem Ablaufdatum zu versehen und abgelaufene Emails automatisch zu löschen ist dem Autor nur beim Emailclient Microsoft Outlook [1] bekannt, der ein solches Feature seit Microsoft Office Outlook 2000 enthält. Diese Funktion kann aber nur genutzt werden, wenn sowohl der Sender als auch der Empfänger des Emails diesen Emailclient verwenden.

Um ein Email mit einem Ablaufdatum zu versehen, muss der Benutzer den Menüpunkt "Optionen" aufrufen. In der Dialogbox (siehe Abbildung 2.1) kann man den Punkt "Nachricht läuft ab nach:" aktivieren und ein Datum sowie eine Uhrzeit auswählen. Diese Eingaben bewirken, dass dem ausgehenden Email das Headerfeld *Expiry-Date* hinzugefügt wird. Zum Beispiel:

```
Expiry-Date: Sat, 21 Jun 2008 12:00:00 +0200
```

Beim Empfänger der Nachricht wird dieses Datum mit der lokalen Zeit verglichen und falls das Ablaufdatum überschritten wurde, wird das Email in der Listansicht als durchgestrichen angezeigt um zu symbolisieren, dass es bereits abgelaufen ist. Dabei werden auch schon gelesene Nachrichten nachträglich noch als abgelaufen markiert, falls die Zeit überschritten wurde.

Der von Microsoft verwendete Headername und das Datum entsprechen den im Standard RFC 2822 [2] vorgeschriebenen Formaten. Die Verwendung des Headerfeldes *Expiry-Date* wurde jedoch in keinem Standard festgelegt, ist somit als Eigenlösung von Microsoft zu betrachten und erklärt auch die Tatsache, dass sich keine kompatible Funktion in anderen Emailclients wieder findet.

Zusammenfassend kann man sagen, dass es zwar eine bestehende Lösung für die in dieser Arbeit geforderten Funktion gibt, es jedoch an einer Standardisierung und Verbreitung in anderen Emailclients fehlt. Dadurch wird die sinvolle Nutzbarkeit stark eingeschränkt.

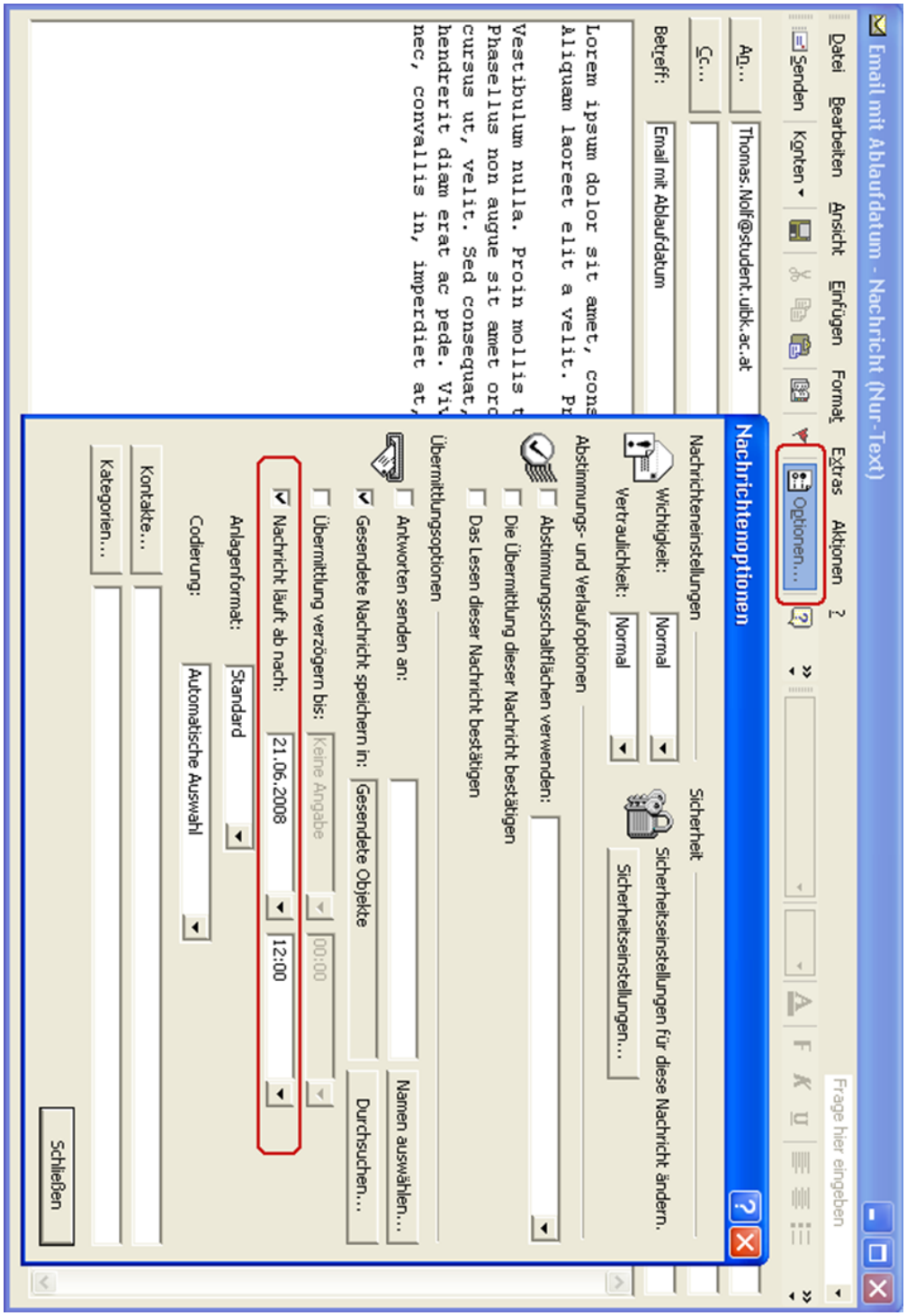

Abbildung 2.1: Ablaufdatum in Microsoft Outlook

3 Lösungsbeschreibung

In diesem Kapitel wird kurz auf Internetstandards und dessen verwaltende Organisation eingegangen. Es wird ein Headerfeld für das Ablaufdatum definiert und erläutert, wie dieses in den Standard eingebracht werden kann. Anschließend wird in den Punkten 3.2 und 3.3 auf die clientseitige und serverseitige Referenzimplementierung eingegangen und deren Funktionsweisen erklärt.

3.1 Einen Standard einbringen

3.1.1 IETF und Standards

Die IETF (Internet Engineering Task Force) ist eine lose und selbstorganisierte Gruppe von Personen, die zur Weiterentwicklung von Internet Technologien beiträgt. Dazu entwickelt sie laufend neue Internet Standards und Dokumente, wie zum Beispiel Protokollstandards, so genannte "Best current practices" und Informationsdokumente in verschiedensten Ausprägungen. Die IETF arbeitet öffentlich, wodurch jede interessierte Person mitarbeiten und seiner Stimme Gehör verschaffen kann. Ihre Mission lautet: "The goal of the IETF is to make the Internet work better."

Alle IETF Standards werden in Form von RFCs (request for comments) publiziert, wobei jedes RFC seinen Ursprung in einem so genannten Internet-Draft hat. Es folgt eine grobe Ablaufbeschreibung zum Einbringen eines IETF Standards um einen Überblick über diesen Prozess zu bekommen:

- Veröffentlichen des Dokuments als Internet-Entwurf (Internet-Draft)
- Sammeln der Kommentare
- Einarbeiten der Kommentare in den Entwurf
- Öfteres Wiederholen der vorigen Punkte
- Einen Area Director bitten den Entwurf bei der IESG (Internet Engineering Steering Group) einzubringen. (Die IESG ist verantwortlich für das technische Management der IETF Aktivitäten)
- Einbringen aller Änderungen, die von der IESG beanstandet wurden. Oft muss das Vorhaben an dieser Stelle auch aufgegeben werden.
- Warten, bis der RFC-Editor das Dokument als RFC veröffentlicht

Eine detaillierte Anleitung zum Einbringen eines IETF-Standards findet sich in [3].

3.1.2 Headerfeld für das Ablaufdatum

Das neue Headerfeld für das Ablaufdatum wurde konform zum Standard RFC 2822 [2] (Internet Message Format) entwickelt, welcher die Syntax von elektronischen Textnachrichten definiert. Das Feld ist von folgender Form:

```
Expires-field = "Expires:" date-time CRLF
```

date-time ist eine Zeitangabe nach Definition *3.3. Date and Time Specification* aus dem Standard [2] mit folgendem Aufbau:

```
date-time       =       [ day-of-week "," ] date FWS time [CFWS]

day-of-week     =       ([FWS] day-name) / obs-day-of-week

day-name        =       "Mon" / "Tue" / "Wed" / "Thu" /
                        "Fri" / "Sat" / "Sun"

date            =       day month year

year            =       4*DIGIT / obs-year

month           =       (FWS month-name FWS) / obs-month

month-name      =       "Jan" / "Feb" / "Mar" / "Apr" /
                        "May" / "Jun" / "Jul" / "Aug" /
                        "Sep" / "Oct" / "Nov" / "Dec"

day             =       ([FWS] 1*2DIGIT) / obs-day

time            =       time-of-day FWS zone

time-of-day     =       hour ":" minute [ ":" second ]

hour            =       2DIGIT / obs-hour

minute          =       2DIGIT / obs-minute

second          =       2DIGIT / obs-second

zone            =       (( "+" / "-" ) 4DIGIT) / obs-zone
```

Es wird im Standard empfohlen, dass für *FWS* jeweils ein einzelnes Leerzeichen verwendet wird. *CFWS* kann ein Kommentar oder ein Leerzeichen sein.

Beispiel eines solchen Headers:

```
Expires: Fri, 29 Aug 2008 00:00:00 +0200
```

3.1.3 Registrierung permanenter Headerfelder

Im Zuge der Recherche zum Thema Email Headerfelder im RFC Repository [13], stellte man fest, dass es ein eigenes IETF-Dokument gibt, das zur Registrierung permanenter Email Headerfelder dient. Dieses RFC mit Nummer 4021 [14] gibt einen guten Überblick über die bereits registrierten Email Header und beinhaltet zwei Einträge, die für das Thema dieser Arbeit relevant sind:

```
2.1.49. Header Field: Expiry-Date

   Description: Message expiry time

   Applicable protocol: Mail

   Status: obsolete

   Author/change controller:
      IETF  (mailto:iesg@ietf.org)
      Internet Engineering Task Force

   Specification document(s):
      RFC 1327

   Related information:
      Time at which a message loses its validity. Introduced by
      RFC 1327 and subsequently changed by RFC 2156 to 'Expires:'.


2.1.50. Header Field: Expires

   Description: Message expiry time

   Applicable protocol: Mail

   Status: standards-track

   Author/change controller:
      IETF  (mailto:iesg@ietf.org)
      Internet Engineering Task Force

   Specification document(s):
      RFC 2156

   Related information:
      Time at which a message loses its validity.
      Renamed version of obsolete Expiry-Date header field.
      RFC 2156 (MIXER), not for general use.
```

Demnach gibt es das in dieser Arbeit geforderte Headerfeld für das Ablaufdatum mit dem Namen *Expires* bereits im Standard. Die weiterführenden Informationen weisen jedoch darauf hin, dass dieses Feld nicht zur generellen Verwendung bestimmt ist, sondern nur für RFC 2156 (MIXER) [15] gilt. MIXER (Mime Internet X.400 Enhanced Relay) definiert die Abbildung zwischen X.400 and RFC 822/MIME - Nachrichten und ist ein Standard zur Konvertierung solcher Nachrichten in einem Email Gateway. Das Feld kann anhand dieser Definition also leider nicht generell in Internet Emails verwendet werden.

Weiters wurde im Zuge der Recherche ein Internet-Draft [16] von Jacob Palme (Stockholm University / KTH, Schweden) aus dem Jahre 1999 gefunden, in dem er unter anderem das Headerfeld *Expires* für die generelle Verwendung in Emails und damit für die Aufnahme in den Standard vorschlägt. In diesem Dokument geht er auch auf das bereits im RFC 4021 vorhandene Feld *Expires* und dessen Verwendung in RFC 2156 (MIXER) ein. Es folgt ein Auszug des Internet-Drafts mit den für diese Arbeit relevanten Textstellen:

```
4. Expires:

Syntax: Expires-field = "Expires:" CFWS date-time [CFWS] CRLF

The Expires header indicates a date-time, at which this
message expires. The field can be used both to limit and to
extend the life of a message. User agents and servers which
employ automatic purging of old messages MAY let this field
influence the purging process. There is no requirement that a
user agent must suppress expired messages or make them
inaccessible to their owners.

Note: This header is also defined, with similar but not
identical meaning, in netnews [8] and in X.420 [4].


5. Relation to X.400 gateways versus Netnews

A similar header to Expires is also defined in
recommendations for gateweying [2] between X.400 [4] and
Internet mail. However, those recommendations are only valid
for gateways. By defining the fields here, the fields are made
available for general Internet e-mail usage. This document
also gives fuller descriptions of the fields than is given by
the X.400 gateweying recommendations [2]. Also an
Auto-Submitted feature has been added to X.420, with similar
definition as in this document.
```

```
Unfortunately, the header Expires has a different name in
Internet-X.400 gatewaying standards [2] and in netnews.

RFC 1327 gives the name "Expiry-Date:" for what in netnews is
called "Expires:" (as specified in RFC 1036).

Because compatibility with netnews is more important than with
X.400, the netnews names of these two fields are proposed
here.

Future versions of RFC 1327 (the MIXER document) may choose to
specify the use of "Expires" also in gatewayed messages from
X.400.
```

Wie aus [17] zu entnehmen, wurde dieses Internet-Draft nie als RFC publiziert und hat den Status "abgelaufen". Auf eine Anfrage gab der Autor des Drafts Auskunft, dass er die Arbeit daran auf finanziellen Gründen abbrechen musste, noch bevor das Draft den in Kapitel 3.1.1 genannten Prozess vollständig durchlaufen konnte.

3.1.4 Standardisierung des Headerfeldes

Dieses Kapitel soll einen Leitfaden zum Einbringen eines neuen Headerfeldes darstellen. Die notwendigen Schritte dazu werden in RFC 3864 [18] genau erläutert und an dieser Stelle nur zusammengefasst. Der Ablauf ist generell in drei Abschnitte unterteilt:

- Erstellen einer Spezifikation für das Headerfeld
- Vorbereiten einer Registrierungsvorlage
- Einreichen der Registrierungsvorlage

Erstellen der Spezifikation

Um ein neues Headerfeld zu registrieren, muss eine Spezifikation erstellt werden, welche die Syntax, Semantik und den Verwendungszweck des Feldes beschreibt. Dabei muss das Feld zumindest die Vorgaben des RFC 2822 in Kapitel 3.6.8 erfüllen und als RFC oder Open Standard [19] veröffentlicht werden.

Vorbereiten der Registrierungsvorlage

Die Registrierungsvorlage muss folgende Form aufweisen (Ausschnitt aus RFC 3864):

```
Header field name:
  The name requested for the new header field. This MUST conform
  to the header field specification of RFC 2822, chapter 3.6.8.
```

```
Applicable protocol:
  Specify "mail" (RFC 2822), "mime" (RFC 2045), "http" (RFC 2616),
  "netnews" (RFC 1036), or cite any other standards-track RFC
  defining the protocol with which the header is intended to be
  used.

Status:
  Specify "standard", "experimental", "informational", "historic",
  "obsoleted", or some other appropriate value according to the
  type and status of the primary document in which it is defined.
  For non-IETF specifications, those formally approved by other
  standards bodies should be labelled as "standard"; others may
  be "informational" or "deprecated" depending on the reason for
  registration.

Author/Change controller:
  For Internet standards-track, state "IETF". For other open
  standards, give the name of the publishing body (e.g., ANSI,
  ISO, ITU, W3C, etc.). For other specifications, give the name,
  email address, and organization name of the primary
  specification author. A postal address, home page URI,
  telephone and fax numbers may also be included.

Specification document(s):
  Reference to document that specifies the header for use with
  the indicated protocol, preferably including a URI that can be
  used to retrieve a copy of the document.  An indication of the
  relevant sections MAY also be included, but is not required.

Related information:
  Optionally, citations to additional documents containing further
  relevant information. (This part of the registry may also be
  used for IESG comments.) Where a primary specification refers
  to another document for substantial technical detail, the
  referenced document is usefully mentioned here.
```

Einreichen der Registrierungsvorlage

Die Registrierungsvorlage muss in eines der "IANA message header field repositories" eingereicht werden. IANA (Internet Assigned Number Authority) [20] ist unter anderem für die Verwaltung von weltweit eindeutigen Namen und Nummern verantwortlich, welche in RFC Dokumenten veröffentlicht wurden und arbeitet sehr eng mit der IETF zusammen. Die Registrierungsvorlage kann auf zwei unterschiedliche Arten eingereicht werden:

- Durch Einfügen eines IANA Hinweis Kapitels im RFC, in welchem man die Registrierung des Headerfeldes fordert. Die Registrierung erfolgt dann

im Zuge des Prozesses der Veröffentlichung des RFC.

- Durch Senden der Vorlage an die zuständige Email Diskussionsliste [21] [22]. Anschließend muss mindestens zwei Wochen auf Diskussionen und Kommentare gewartet werden, bevor man die Vorlage an die IANA unter der Adresse [23] sendet. IANA veröffentlicht die Vorlage, wenn das Dokument ihren Kriterien entspricht und die IESG (Internet Engineering Steering Group) oder deren zuständige Fachkundige keinen Einspruch erheben.

3.2 Referenzimplementierung

Ziel dieser Arbeit war es auch eine Referenzimplementierung vorzunehmen, anhand derer die Funktionsweise und der praktische Nutzen demonstriert werden kann. Anforderung war, einen bestehenden Emailclient mit einer Funktionalität zu erweitern, die es den Benutzern ermöglicht auf einfache Art und Weise ein Email mit einem Ablaufdatum zu versehen. Natürlich sollte diese Erweiterung auch möglichst einfach zu installieren sein um potentielle Benutzer nicht durch eine komplexe Installation abzuschrecken.

Um eine möglichst große Anzahl an Benutzern bedienen zu können fiel die Entscheidung auf den Open-Source Emailclient Mozilla Thunderbird [4]. Dieser Emailclient basiert auf dem Quelltext von SeaMonkey [5] (früher: Mozilla Application Suite), ist für verschiedenste Betriebssysteme verfügbar und erfreut sich großer Beliebtheit. Durch das Extension-System können Mozilla-Applikationen beliebig um zusätzliche Funktionen erweitert werden, ohne deren Quellcode selbst ändern zu müssen. Das Installieren von Extensions erweist sich als einfach und auch eine Update-Funktion für bereits installierte Extensions ist vorhanden.

3.2.1 Grundlagen einer Mozilla Extension

Wie schon erwähnt ist eine Mozilla Extension eine Erweiterung einer Mozilla-Applikation (Firefox oder Thunderbird), die zusätzliche Funktionalität bietet. Aus technischem Gesichtspunkt gibt es keinerlei Unterschied zwischen einer Extension für Firefox und Thunderbird, weshalb folgende Informationen für beide gleichermaßen gelten.

Jede Mozilla Extension ist in folgende drei Teile aufgeteilt: Die Struktur, das Aussehen und das Verhalten.

1. Teil: Struktur

Der Strukturteil bestimmt die grafischen Elemente einer Extension wie Menüs, Buttons, etc. sowie deren Positionierung und wird in Mozilla's XML-basierter Sprache XUL (XML User Interface Language) beschrieben. Dabei ist die gesamte Oberfläche des Emailclients durch verschiedene XUL-Dateien beschrieben, wobei jede Komponente eine eindeutige ID aufweist. Durch die so genannte

Overlay Technik von Mozilla können vorhandene Komponenten durch das Bereitstellen eigener XUL-Dateien erweitert werden ohne im bestehenden Quellcode Änderungen vornehmen oder die komplette Oberfläche neu beschreiben zu müssen.

Folgende XUL-Datei erweitert die Toolbar um einen zusätzlichen Button:

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/css"
  href="chrome://expirydate/content/expiryDate.css"?>

<script type="application/x-javascript"
  src="chrome://expirydate/content/expiryButton.js"/>

<!DOCTYPE overlay>
<overlay id="expirydate-overlay"
 xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<toolbarpalette id="MsgComposeToolbarPalette">
  <toolbarbutton id="expiry-button-1"/>
</toolbarpalette>

<toolbarbutton id="expiry-button-1"
  label="Expiry-Date" tooltiptext="Expiry-Date" class="StyleButton1"
  oncommand="doSomething()" />
</overlay>
```

Interessant sind hier die letzten drei Blöcke. Der erste Block gibt an, dass es sich bei dieser XUL-Datei um ein Overlay handelt. Die beiden anderen Blöcke erweitern die Toolbar mit der ID *MsgComposeToolbarPalette* um einen neuen Button mit der Bezeichnung Expiry-Date.

2. Teil: Aussehen

In den XUL-Dateien werden den grafischen Komponenten üblicherweise Klassennamen zugewiesen um dessen Aussehen mittels Cascading Stylesheets [6] anpassen zu können. Mit folgender Zeile werden in voriger XUL-Datei Stylesheetinformationen importiert:

```
<?xml-stylesheet type="text/css"
  href="chrome://expirydate/content/expiryDate.css"?>
```

Die Klassenvergabe erfolgt wie in HTML indem man das Attribut *class="Klassenname"* dem jeweiligen Tag hinzufügt, wie es im Beispiel-XUL weiter oben im letzten Block beim Button gemacht wurde.

3. Teil: Verhalten

Dieser Teil bestimmt die eigentliche Funktionalität der Extension. Jegliche Aktionen und Programmabläufe müssen in Form von JavaScript Funktionen bereitgestellt werden, welche mit dem Struktur-Teil verbunden und davon auch aufgerufen werden. Dieses Aufrufen basiert auf dem Event-Konzept von Mozilla. Mausklicks, Tastatureingaben und andere Interaktionen des Benutzers mit der Oberfläche, den so genannten GUI-Events, führen zum Aufruf einer JavaScript-Methode, welche im XUL-Teil der jeweiligen Komponente definiert wurde. In diesem Beispiel wird durch den Teil *oncommand="doSomething()"* bei Anklicken des Buttons die Methode *doSomething()* ausgeführt. Das JavaScript, welches diese Methode enthält, wird durch folgende Zeile importiert:

```
<script type="application/x-javascript"
  src="chrome://expirydate/content/expiryButton.js"/>
```

3.2.2 Aufbau der Extension Expiry-Date

Die in dieser Arbeit entwickelte Extension liegt in gepackter Form mit Namen Expiry-Date.xpi vor. XPI (Cross-Platform Installation) ist jene Technologie, die von Mozilla zur plattformunabhängigen Installation von Applikation eingesetzt wird. Es handelt sich dabei um ein ZIP-File, welches neben den eigentlichen Dateien noch für die Installation relevante Beschreibungsdateien enthält.

Wenn man die Datei in Expiry-Date.zip umbenennt und mit einem Zip-Programm entpackt, erkennt man die Struktur wie sie in Abbildung 3.1 zu sehen ist. In Tabelle 3.2.2 findet sich eine kurze Beschreibung der jeweiligen Dateien.

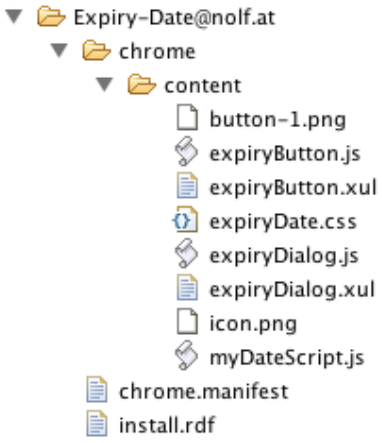

Abbildung 3.1: Struktur der entwickelten Extension

3.2.3 Funktionsweise der Extension

Die Extension wird zum ersten mal geladen, wenn ein Benutzer auf den Button zum Erstellen eines neuen Emails klickt. Zu diesem Zeitpunkt werden auch zwei

Dateiname	Teil	Beschreibung
button-1.png	-	Enthält die Bilder der Buttons
expiryButton.js	Verhalten	Beinhaltet Funktionen zum Öffnen des Dialogfensters und setzen des Headers
expiryButton.xul	Struktur	Beinhaltet den neuen Toolbarbutton
expiryDate.css	Aussehen	Legt das Aussehen fest
expiryDialog.js	Verhalten	Verarbeitet die Eingabe des Datum
expiryDialog.xul	Struktur	Beinhaltet das Dialogfenster inkl. Eingabefeld
icon.png	-	Icon der Extension
myDateScript.js	Verhalten	Implementiert die laufende Überprüfung der Benutzereingabe
chrome.manifest	-	Enthält das Mapping von internen Namen zu physichen Pfaden
install.rdf	-	Vom Extension-Manager zur Installation benutzt. Enthält auch Name, Author, etc.

Tabelle 3.1: Beschreibung der einzelnen Dateien

Listener beim Objekt *msgComposeWindow* registriert. Der erste Listener sorgt dafür, dass eine Methode zum Initialisieren der Extension aufgerufen wird. Der zweite hört auf den Event mit dem Namen *compose-send-message*, welcher kurz vor Absenden eines Emails an alle Listener versendet wird und hier die Methode *nolf_addExpiryDateBeforeSend()* aufruft. Wie später noch genauer erläutert wird sorgt diese Methode dafür, dass das eingegebene Ablaufdatum in ein korrektes Header-Format gebracht und im Email mitgesendet wird.

Beim Klick auf den Button "Expiry-Date" (siehe Abbildung 3.2) wird die Methode *nolf_openExpiryDialog()* aufgerufen, welche die in der Datei *expiryDialog.xul* definierte Struktur in Form einer Dialogbox öffnet. Wie in Abbildung 3.3 zu sehen ist, handelt es sich dabei um ein einfaches Dialogfenster mit einer Bezeichnung 'Expiry-Date' und einem Eingabefeld, welches die Eingabe eines Datums im Format "yyyy/mm/dd" (Jahr/Monat/Tag) erwartet. Gleich beim Öffnen des Dialoges wird ein Listener auf das Eingabefeld gelegt, welcher auf das Event *keyUp* hört. Immer dann, wenn ein Benutzer ein einzelnes Zeichen eingegeben hat, löst dieser Event den Aufruf der Methode *nolf_checkUserInputTextbox()* aus. Diese Methode versucht bereits während einer aktiven Benutzereingabe das bisher Eingegebene auf syntaktische und semantische Korrektheit hin zu überprüfen. Wird ein Fehler festgestellt, so wird die Schriftfarbe der Textbox auf Rot gesetzt um dem Benutzer ein Problem bei der Eingabe zu signalisieren, noch bevor er diese abgeschlossen hat (Abbildung 3.4).

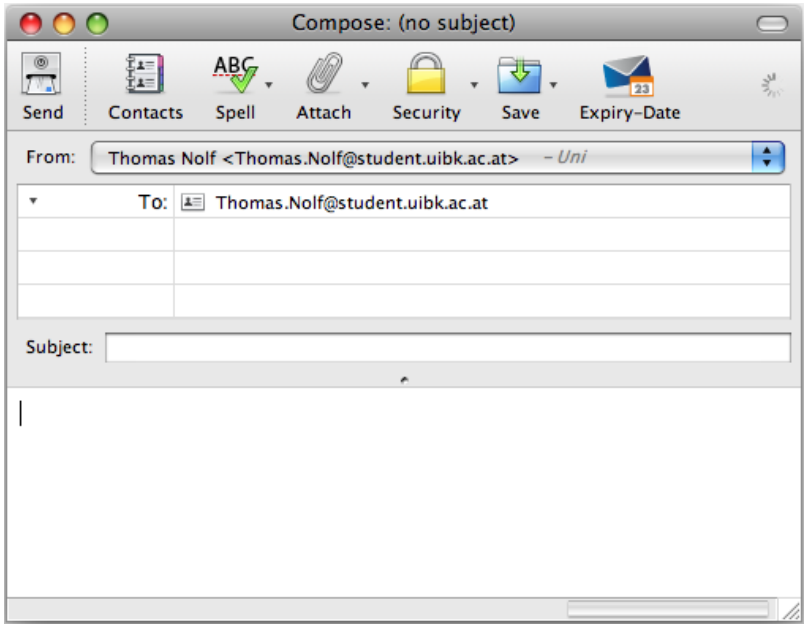

Abbildung 3.2: Button Expiry-Date (rechts oben)

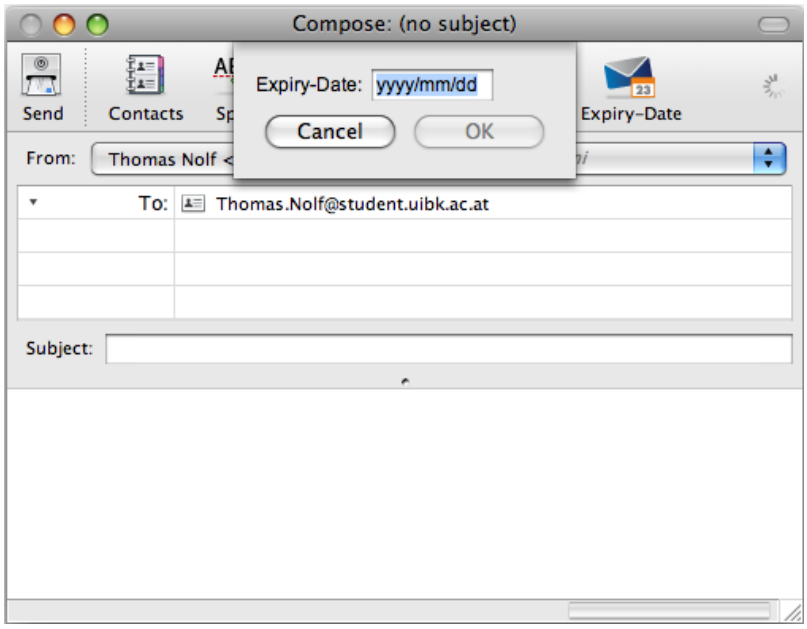

Abbildung 3.3: Dialogfenster zum Erfassen des Ablaufdatums

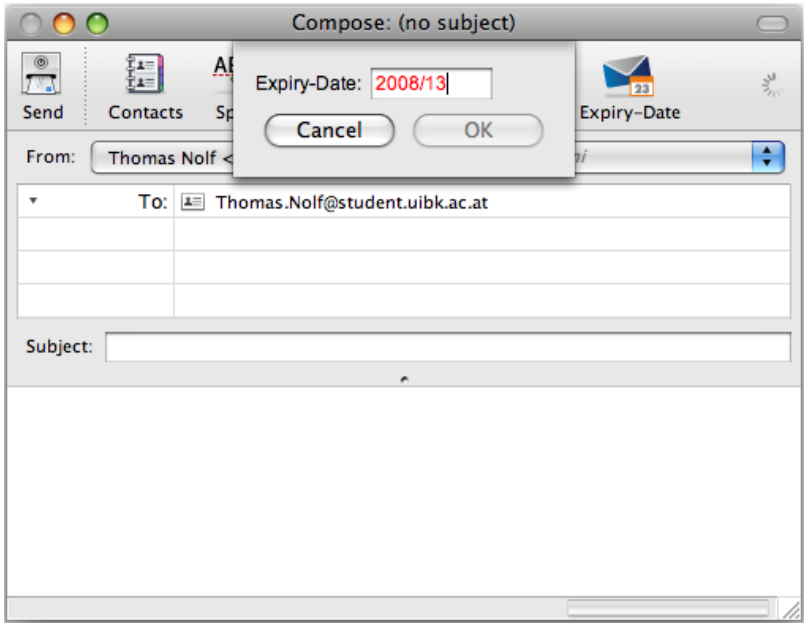

Abbildung 3.4: Eingabe eines falschen Datums

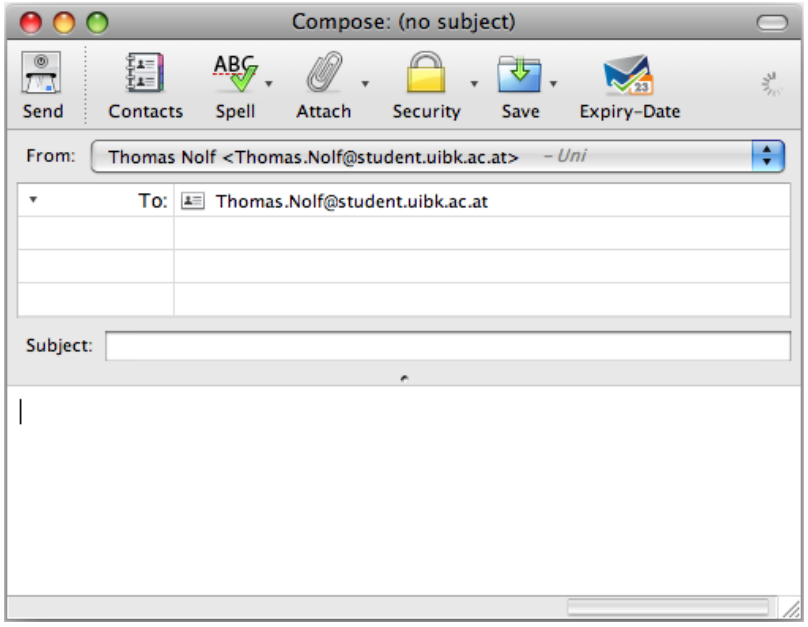

Abbildung 3.5: Korrekt eingegebenes Ablaufdatum

Erst wenn alle 10 Zeichen eingegeben wurden und das Datum syntaktisch sowie semantisch korrekt ist, wird der Bestätigungsbutton aktiviert und damit für den Benutzer zum Abschluss seiner Eingabe freigegeben. Dies passiert durch Setzen des Button-Attributs *disabled=false*.

Durch Anklicken des Buttons "OK" oder Drücken der Taste "Enter" wird die Methode *nolf_accept()* aufgerufen, welche das eingegebene Datum in einer globalen Variable speichert, das Dialogfenster wieder schließt und mittels CSS einen anderen Button in der Toolbar anzeigt, der signalisiert, dass bereits ein Datum eingegeben wurde (Abbildung 3.5). Natürlich kann das Dialogfenster erneut geöffnet und das Datum verändert oder auch wieder gelöscht werden. Entsprechend wird die globale Variable verändert bzw. initialisiert und der Button in der Toolbar aktualisiert, sodass er den richtigen Status anzeigt.

Nach Fertigstellen des Emails und Absenden durch den Benutzer löst das Objekt *msgComposeWindow* den eingangs beschriebenen Event *compose-send-message* aus, für den sich die Extension mit der Methode *nolf_addExpiryDate-BeforeSend()* als Listener registriert hat. Diese Methode liest das eingegebene Datum aus der globalen Variable, konvertiert es in ein RFC 2822 konformes Datum und befüllt damit das neue Headerfeld *Expires*, welches an entsprechender Stelle im globalen Array von Thunderbird gesetzt wird. Anschließend übernimmt Thunderbird intern das Zusammenstellen und Versenden des Emails.

3.3 Serverseitige Implementierung

Die serverseitige Implementierung dieser Arbeit wurde aufgrund der starken Synergien als Erweiterung der Bakkalaureatsarbeit *Benutzerunterstützte Email-reduktion* [7] entwickelt, in welcher die beiden Programme *Vacation* und *SpontaneousRecall* entwickelt wurden. Neben diesen beiden "Hauptklassen" entstanden auch Hilfsklassen, welche als Basis für die serverseitige Implementierung dieser Arbeit verwendet wurden und bereits eine große Anzahl von Funktionalität bereitstellten.

3.3.1 Anforderungen

Das Programm *Expiry* soll sowohl auf einem entfernten Rechner, als auch direkt auf dem Mailserver ausgeführt werden können, wobei es manuell gestartet und einmalig oder in regelmäßigen Abständen laufen soll. Die Aufgabe des Programmes ist es, in der Mailbox nach abgelaufenen Emails zu suchen und diese zu löschen bzw. als abgelaufen zu markieren. Dabei soll es jenes Ablaufdatum finden, das der Benutzer im definierten Headerfeld (*"Expires"*) oder auch im Betreff des Emails in der Form "Expires: yyyy/mm/dd" gesetzt hat und entsprechend verarbeiten.

3.3.2 Zielumgebung und Technologien

Als Emailserver wurden die beiden Rechner mail.uibk.ac.at und mail2.uibk.ac.at der Universitätsserver Innsbruck definiert. Es handelt sich dabei um Unix-Server mit POP3-Zugang, installiertem Java Runtime Environment in Version 1.5 und möglichem SSH-Zugriff. Das Programm *Expiry* soll aber auch auf beliebigen, entfernten Rechnern mit installierter Java Runtime größer 1.5 verwendet werden können.

Die eingesetzten Technologien wurden weitestgehend schon durch die Entscheidung zur Erweiterung des Paketes *EmailReduction* [7] festgelegt. So kommen auch bei dieser Arbeit die plattformunabhängige Programmiersprache **Java** [8], die **JavaMail API** [9], sowie **log4j** [10] und **JUnit** [11] zum Einsatz. Die Einsatzgebiete dieser Technologien werden in Kapitel 3.3 der Bakkalaureatsarbeit *Benutzerunterstützte Emailreduktion* [7] kurz erläutert und können dort nachgelesen werden.

3.3.3 Struktur der Erweiterung

Die Struktur des Projekts *EmailReduction* wurde nicht verändert, sondern lediglich erweitert. Es wurde eine neue, ausführbare Hauptklasse *Expiry.java* im Paket *at.nolf* hinzugefügt, welche den generellen Ablauf und die Logik der Erweiterung enthält. Außerdem wurde dem Paket *at.nolf.common* die neue Exception *DateException.java* hinzugefügt, welche einen Fehler beim Parsen des Ablaufdatums aus dem Betreff oder dem Header signalisiert. Die zusätzliche Testklasse *ExpiryTest.java* im Paket *at.nolf.test* testet das Finden, Extrahieren und Prüfen des Ablaufdatums in Form von JUnit Testfällen. Abbildung 3.6 gibt einen Überblick über die komplette Struktur des Quellcodes.

3.3.4 Umsetzung der Funktionalität zum Löschen abgelaufener Emails

Der Ablauf von *Expiry* stellt sich wie folgt dar: nach Start der Anwendung wird in der Methode *init()* über die Klasse *MailHelper* eine Verbindung zum Postfach hergestellt und die Konfigurationsdatei expiry.properties eingelesen. Im Anschluss daran wird die Hauptmethode *runExpiry()* gestartet, in welcher zu Beginn alle Emails gesucht werden, die entweder ein Headerfeld mit Namen *"Expires"* oder dieses Wort im Betreff enthalten. Dieser Suchvorgang wurde mit den so genannten SearchTerms der JavaMail API implementiert um ein schnelles Suchen innerhalb der Mailbox zu ermöglichen ohne alle Emails zuerst komplett übertragen zu müssen. Mithilfe der SearchTerms werden zur Suche nämlich nur die relevanten Teile der Emails wie zum Beispiel Betreff oder Headerfelder übertragen. An diesem Punkt kann durch Konfiguration auch festgelegt werden, dass immer alle Emails der Mailbox durchsucht werden sollen oder nur jene, die seit dem letzten Durchlauf neu hinzu gekommen sind (siehe Readme).

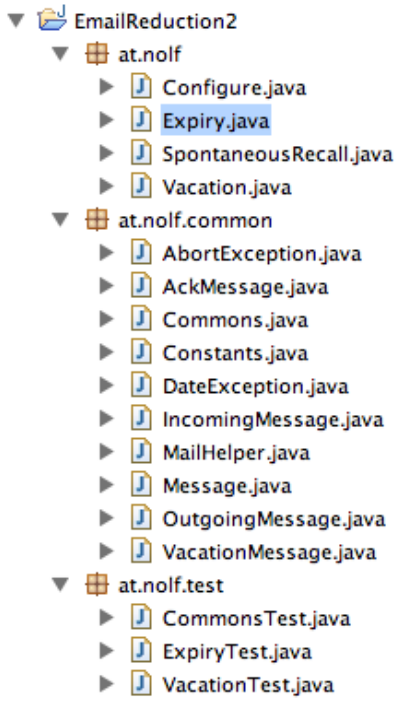

Abbildung 3.6: Struktur der serverseitigen Implementierung

Anschließend wird für jedes gefundene Email eruiert, ob sich das Ablaufdatum im Headerfeld oder Betreff befindet und je nachdem spezifisch gehandelt:

Ablaufdatum in Headerfeld

Nachdem das Datum definitionsbedingt als String im RFC 2822 konformen Format vorliegt, muss dieser String nur mehr in ein entsprechendes Datumsobjekt konvertiert werden. Hierfür bietet sich die Klasse *MailDateFormat* der JavaMail API an, welche mittels der Methode *parse(String)* einen RFC2822-String einliest und daraus ein *Date*-Objekt erzeugt. Dies passiert durch folgende zwei Zeilen:

```
MailDateFormat mdf = new MailDateFormat();
Date expiryDate = mdf.parse(expiryDateString);
```

Anschließend wird das daraus gewonnene Datum mit dem aktuellen verglichen. Ist es kleiner, so ist das Email bereits abgelaufen und kann entsprechend der Konfiguration gelöscht oder als abgelaufen markiert werden.

Ablaufdatum im Betreff

Befindet sich das Datum im Betreff, so muss vorher der relevante Teil daraus extrahiert werden. Dies passiert in der Methode *extractDateFromSubject(String subject)* mittels Patternmatching, wobei folgender regulärer Ausdruck auf den Betrefftext angewendet wird:

```
"Expires: [0-9]{4}/[0-9]{2}/[0-9]{2}" // zB. Expires: 2008/09/28
```

Die Teile des Datums müssen dann einzeln geparst, auf semantische Korrektheit geprüft und in ein Datums-Objekt gepackt werden, bevor es verglichen werden kann. Ein Vergleich mit dem aktuellen Datum entscheidet wieder, wie mit dem Email weiter vorgegangen werden soll. Ein Beispiel eines gültigen Betreffs mit Ablaufdatum findet sich in Abbildung 3.7.

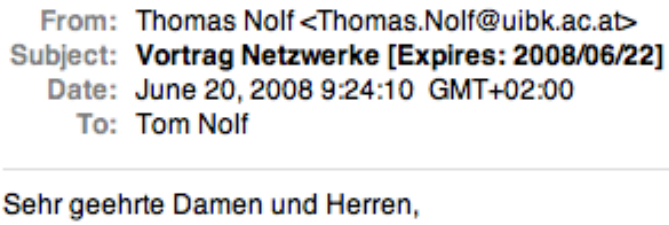

Abbildung 3.7: Auszug eines Emails mit Ablaufdatum im Betreff

3.3.5 Ausführen des Programmes Expiry

Das Programm *Expiry* kann gestartet werden, indem einer der folgenden Befehle in einer Kommandozeile ausgeführt wird:

```
java -cp EmailReduction.jar at.nolf.Expiry
```

Oder wenn das Programm als Dämon in bestimmten Zeitintervallen laufen soll:

```
java -cp EmailReduction.jar at.nolf.Expiry daemon
```

Das Zeitintervall des Dämons kann in der Konfigurationsdatei *expiry.properties* festgelegt werden und ist standardmäßig auf fünf Minuten eingestellt.

4 Test der Lösung

Um die Funktionsweise der jeweiligen Implementierung bestmöglich überprüfen zu können, wurden die Tests in drei unterschiedliche Bereiche aufgeteilt. Dabei bestehen die ersten beiden Bereiche aus Einzeltests der Client- und der Servererweiterung und wurden so weit als möglich durch automatisierte Testläufe abgedeckt. Der dritte Bereich stellt einen Gesamttest dar und überprüft das korrekte Zusammenspiel.

4.1 Test der Thunderbird Extension

Die Tests begannen mit der Installation der Extension auf einer frischen Thunderbird Installation und Überprüfung der Grundfunktionen, wie dem Öffnen des Dialogfensters und dem Wechsel des Buttons nach Eingabe eines korrekten Datums oder nach Abbruch bzw. Löschen des eingegebenen Datums.

Ein wichtiger Punkt stellte das Testen der laufenden Überprüfung der Benutzereingabe dar, weil es hier sehr viele verschiedene zu berücksichtigende Konstellationen gibt. Es musste sowohl die syntaktische als auch semantische Korrektheit der meist noch unvollständigen Benutzereingaben bestmöglich überprüft werden. Für diese Testfälle war es wichtig ein Testframework einzusetzen, damit die Funktionsweise nach Änderungen am Programm immer wieder schnell sichergestellt werden konnte.

4.1.1 JsUnit für automatisierte JavaScript Tests

JsUnit [12] ist eine Portierung des bekannten Java Testframeworks JUnit und damit ein Open Source Unit Testing Framework für clientseitiges JavaScript. Mithilfe dieses Frameworks war es möglich eine große Anzahl verschiedener Testfälle in der Datei *myDateScriptTest.html* zu sammeln und innerhalb des Frameworks laufen zu lassen. Zum Beispiel beinhaltet diese Sammlung Fälle mit verschiedenen, syntaktischen Fehleingaben des Benutzers, weiters mit semantischen Fehlern wie zum Beispiel Überschreitung der Tage zu einem bestimmten Monat (z.B.: 31. Juni), Berücksichtigung von Schaltjahren und auch Tests für die Funktion zum Umwandeln eines Datums in einen RFC 2822 konformen Datum-String.

4.2 Test der serverseitigen Erweiterung

Teile der serverseitigen Implementierung wurden mit automatisierten Tests und dem JUnit-Framework getestet. Vor allem das Finden und Extrahieren des Ablaufdatums aus dem Betreff und die Überprüfung der Syntax und Semantik von verschiedenen Fällen wurde dadurch überprüft.

Weiters wurde die Erweiterung auch manuellen Tests mit "echten" Emails und verschiedenen Ausprägungen des Ablaufdatums im Betreff sowie unterschiedlichen Konfigurationen unterzogen und die Ausgabe in Log- und Statistikdateien kontrolliert.

4.3 Gesamttest

Ein abschließender Gesamttest stellte sicher, dass die beiden Lösungen miteinander kompatibel sind und der Gesamtablauf funktioniert. Hier wurde vor allem Augenmerk auf die Variante von Emails mit Ablaufdatum im Headerfeld gelegt und ausgiebig getestet.

5 Zusammenfassung

5.1 Allgemeines

Folgende Software wurde bei dieser Arbeit verwendet:

- Mozilla Thunderbird 2.0.0.14 für Mac OS X
- Mozilla Add-on: Extension Developer 0.3.0.20060726
- Mozilla Add-on: DOM Inspector 1.8.1.2
- Eclipse 3.2.2 für Mac OS X
- Fat Jar Eclipse Plugin 0.0.27 (Zum Erzeugen der ausführbaren Jar-Datei)
- TeXShop 2.10 (Zum Erstellen dieses Dokuments in LaTeX)

Folgende Ordnerstruktur befindet sich auf der Programm-CD:

- *source/client* (Quellcode der Thunderbird Extension)
- *source/server* (Quellcode der serverseitigen Erweiterung als Eclipse-Projekt)
- *target/client* (Thunderbird Extension im gepackten xpi-Format)
- *target/server* (Ausführbare Jar-Datei und Konfigurationsdateien)
- *documents* (Dokumentation der Arbeit im Latex- und PDF-Format, sowie Javadoc des Quellcodes im Html-Format)

5.2 Rückblick und Danksagung

In dieser Arbeit wurde eine weitere Möglichkeit zur benutzerunterstützten Reduktion von Emails entwickelt. Dazu wurde ein Headerfeld definiert, welches konform zum Standard RFC 2822 [2] ist und als Ablaufdatum eines Emails fungiert. Anschließend wurde auf die Standards der IETF eingegangen und beschrieben, welche Schritte notwendig wären um dieses neue Headerfeld in den Standard aufgenommen zu bekommen.

Weiters wurde in dieser Arbeit eine Referenzimplementierung vorgenommen, um die Funktionsweise zu demonstrieren. Dazu wurde einerseits eine Erweiterung für den Mozilla Thunderbird Emailclient entwickelt, die es den Benutzern erlaubt auf einfache Art und Weise ein Ablaufdatum zu setzen. Andererseits

wurde eine Lösung entwickelt, die sich um die serverseitige Löschung abgelaufener Emails kümmert. Den Abschluss bildeten Testreihen und die Erstellung dieses Dokuments.

Abschließend möchte ich mich noch bei meinen beiden Betreuern Dr. Michael Welzl und Dr. Barbara Weber für die Unterstützung bedanken.
Vielen Dank!

Literaturverzeichnis

[1] Microsoft Outlook:
http://office.microsoft.com/en-us/outlook/HA010917601033.aspx
(Letzter Zugriff: 06.05.2008)

[2] Internet Message Format:
http://www3.tools.ietf.org/html/rfc2822 (Letzter Zugriff: 17.06.2008)

[3] The Internet Standards Process:
ftp://ftp.isi.edu/in-notes/rfc2026.txt (Letzter Zugriff: 17.06.2008)

[4] Mozilla Thunderbird:
http://www.mozilla.com/en-US/thunderbird/ (Letzter Zugriff: 17.06.2008)

[5] SeaMonkey:
http://www.seamonkey-project.org/ (Letzter Zugriff: 17.06.2008)

[6] Cascading Stylesheets:
http://en.wikipedia.org/wiki/Cascading_Style_Sheets
(Letzter Zugriff: 18.06.2008)

[7] Benutzerunterstützte Emailreduktion:
http://welzl.at/research/tools/email/index.html
(Letzter Zugriff: 20.06.2008)

[8] Java: *http://java.sun.com/javase/* (Letzter Zugriff: 11.05.2008)

[9] JavaMail API: *http://java.sun.com/products/javamail/*
(Letzter Zugriff: 25.04.2008)

[10] Apache log4j: *http://logging.apache.org/log4j/* (Letzter Zugriff: 14.02.2008)

[11] JUnit: *http://www.junit.org* (Letzter Zugriff: 14.02.2008)

[12] JsUnit: *http://www.jsunit.net/* (Letzter Zugriff: 21.06.2008)

[13] RFC Repository: *http://www.ietf.org/rfc.html* (Letzter Zugriff: 22.06.2008)

[14] Registration of Mail and MIME Header Fields:
http://www3.tools.ietf.org/html/rfc4021 (Letzter Zugriff: 22.06.2008)

[15] MIXER (Mime Internet X.400 Enhanced Relay):
http://www3.tools.ietf.org/html/rfc2156 (Letzter Zugriff: 22.06.2008)

[16] The Auto-Submitted and Expires Headers in E-mail:
http://tools.ietf.org/html/draft-ietf-mailext-new-fields-15
(Letzter Zugriff: 23.06.2008)

[17] Status of draft-ietf-mailext-new-fields-15:
https://datatracker.ietf.org/drafts/draft-ietf-mailext-new-fields/
(Letzter Zugriff: 23.06.2008)

[18] Registration Procedures for Message Header Fields:
http://www3.tools.ietf.org/html/rfc3864 (Letzter Zugriff: 24.06.2008)

[19] The Internet Standards Process – Revision 3:
http://www3.tools.ietf.org/html/rfc2026#section-7
(Letzter Zugriff: 24.06.2008)

[20] Internet Assigned Numbers Authority (IANA):
http://iana.org (Letzter Zugriff: 24.06.2008)

[21] Emailadresse zum Ankündigen neuer Headerfeld-Einbringungen:
ietf-message-headers@lists.ietf.org

[22] Emailadresse zum Beitreten der Mailingliste ietf-message-headers:
ietf-message-headers-request@lists.ietf.org

[23] Emailadresse zur Einreichung neuer Headerfeld-Vorlagen:
iana@iana.org

6 Anhang: Readme

Email Reduction through SpontaneousRecall, Vacation & Expiry

Version 1.1
Release date: 2008-06-26

by Tom Nolf

University of Innsbruck
thomas.nolf@student.uibk.ac.at

1 Introduction

You just downloaded a Java package that was developed to give people the possibility to reduce the amount of emails by recalling messages or by providing an expiry date. Although there is only one jar-file there are three independently working applications, namely **Vacation**, **SpontaneousRecall** and **Expiry**.

1.1 Vacation

at.nolf.Vacation was designed to run directly on a mailserver. As its name suggests, this program is capable of responding to emails with vacation messages. *Vacation* includes a random key in the vacation message so that people can simply reply to it in order to recall their initial email. *Vacation* uses an email template to create vacation messages and does only respond with a vacation message to the same sender within a configurable time interval.

1.2 SpontaneousRecall

at.nolf.SpontaneousRecall enables users to recall any email as long as it hasn't already been fetched. People can resend a sent email marking it with a special label in the subject, which is then called a recall email. When receiving such a recall email, *SpontaneousRecall* tries to find the initial appropriate email by applying various algorithms. Once it has found exactly one match the email gets either deleted or marked as recalled to be filtered by rules of various email clients. The behaviour is configurable.
SpontaneousRecall can operate in three different ways. It can be instantiated and run directly on the mailserver, started manually on a remote machine to run once, or run as a daemon on a remote machine in a configurable time interval.

1.3 Expiry

at.nolf.Expiry is capable of finding and deleting emails in your mailbox that are expired. People can provide expiry dates in a headerfield or in the subject of

their emails that define the duration of validity. When this duration is expired the email gets deleted or marked as expired by this program, which can be run directly on the mailserver or on a remote machine.

Application	Utilization	Runs on
Vacation	Recall emails by replying to a vacation message	Mailserver
SpontaneousRecall	Recall any email	Mailserver or Remote Machine
Expiry	Deletes expired emails	Mailserver or Remote Machine

Table 1. Overview of Utilization

2 Installing, Configuring and Using it

2.1 Vacation

Example Sequence for Understanding
After someone sends you an email (s)he gets an automatic vacation message where you may suggest the possibility to recall his email. In order to recall the initial email the person simply has to reply to the vacation email. If (s)he additionally wants to get an acknowledgement, (s)he has to insert the label *ACK* at the very beginning of the subject when replying to the vacation email.

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. Just upload the jar-file into the root directory of your mailserver and edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *Vacation* upon arrival of new email. This can be achieved by copying the following snippet in your .procmailrc before the part where the email gets delivered to INBOX:

```
# comment those 3 lines if you are not on vacation
:0ic
```

```
*
| java -cp EmailReduction.jar at.nolf.Vacation

:0B
* .*\<RECALL-1\>thomas.nolf@student.uibk.ac.at\</RECALL-1\>
| java -cp EmailReduction.jar at.nolf.Vacation recall
```

The first three lines after the comment call *Vacation* upon each arrival of email so that it can send a vacation message if necessary. You should comment those 3 lines with the character '#' if you are not on vacation.

The second block is responsible for recognizing a reply to a vacation message and starting *Vacation* in recall mode. You should put your email address between the tags, which you specified in the configuration file connection.properties (key: mail.smtp.from, see chapter 2.8 for details).

Template for Vacation message
Since *Vacation* uses a template to create a vacation message, one can create and modify it ad libitum. The template must be a text file named vacation.msg which has to be located in the same directory as the jar-file. Here is the content of my vacation.msg:

```
From: Thomas Nolf <thomas.nolf@student.uibk.ac.at>
Subject: Urlaub/vacation (Re: $SUBJECT)
Precedence: bulk

I am on vacation until xxxx.
Please refer all urgent business to xxx@uibk.ac.at
Tom Nolf

######################################################################
# You can recall your message (delete it from my mailbox)
# as long as I haven't read it.
# To do so, simply reply to this message without altering the body.
# $RECALL-LINE-1
# $RECALL-LINE-2
######################################################################
```

Explanation: It is important to know that all lines until the first empty line are interpreted as header fields for the vacation message. You can add any header field according to the specification (RFC 2822).

$SUBJECT and the lines $RECALL-LINE-X are placeholders that are substituted by the application. $SUBJECT is replaced by the subject of the initial email, $RECALL-LINE-1 is substituted with the generated random key

that makes it possible to delete the initial email by simply replying to it and $RECALL-LINE-2 will contain your email address to uniquely assign the message. Both $RECALL-LINE-1 and $RECALL-LINE-2 are mandatory in order to allow recalling of emails.

Template for Confirmation messages
Vacation (and also *SpontaneousRecall*) uses templates to create the confirmation messages about the successful or unsuccessful recall. Both templates must be text files in the same format as vacation.msg and named ack_positive.msg and ack_negative.msg respectively. One can use the placeholder $SUBJECT in these templates which will be replaced with the subject of the recall email. Here is the content of my ack_negative.msg:

```
Precedence: bulk
X-No-Archive: yes
Subject: Recall of $SUBJECT was not successful

Your recall of the message with the subject $SUBJECT was not
successful. Most likely the email has been fetched already.
```

Configuration
Vacation needs to know a few things about your mailaccount to work accordingly and expects this information in the file connection.properties. This file can easily be created using the tool *at.nolf.Configuration* or manually. See chapter 2.6 about how to create a configuration with this tool and chapter 2.8 to see all possible configurations.

Logs and Statistics
Detailed logs of *Vacation* are written to the file recall.log, but it also writes less detailed statistical information to a separate file called vacation_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.2 SpontaneousRecall

Example Sequence for Understanding
Imagine someone sends you an email and wants that to be undone as long as you haven't fetched it. In this case the person has to forward the sent message without altering its content and indicate it as a recall email by specifying a special recall label (i.e. *RECALL-ME*) at the very beginning of the subject. If (s)he additionally wants to get a confirmation of the recall, (s)he has to append the label *-ACK* to the defined recall label when forwarding the email.

Example:

```
Subject of initial email: Hello World
Subject of recall email: RECALL-ME Hello World
Subject of recall email with confirmation: RECALL-ME-ACK Hello World
```

General Configuration
SpontaneousRecall needs to know a few things about the mailaccount that it should handle. It shares this information with *Vacation* and therefore reads all needed configurations from the file connection.properties. Depending on whether *SpontaneousRecall* runs directly on the mailserver or on a remote machine, the hostnames may slightly differ (ie. localhost, mail2.uibk.ac.at).

Moreover the behaviour of *SpontaneousRecall* needs to be configured by providing a file named recall.properties. There you can define if a recalled email gets deleted or just marked, the label that should be specified in the subject to recall an email, and the algorithms that should be applied to find the initial email to recall.

Both configuration files can easily be created by using the tool *at.nolf.Configuration* or even manually. See chapter 2.6 about how to create a configuration with this tool and chapter 2.8 to see all possible configurations.

Templates for Confirmation Messages
See chapter 2.1 (Templates for confirmation messages) for details.

Logs and Statistics
Detailed logs of *SpontaneousRecall* are written to the file recall.log, but it also writes less detailed statistical informations to a separate file recall_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.3 SpontaneousRecall on Mailserver

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. If you didn't already upload the files for *Vacation*, do it now. Afterwards edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *SpontaneousRecall* on arrival of email. This can be achieved by copying the following snippet to your .procmailrc **after** the part where the email gets delivered to INBOX. It's very important that *SpontaneousRecall* is run *after* the email was delivered to your mailbox, because it searches and deletes the email directly from your mailbox and therefore the email has to be in your INBOX before running.

```
:0wc
| $DELIVER +INBOX

:0Wi
* ^Subject:.*RECALL-ME.*
| java -cp EmailReduction.jar at.nolf.SpontaneousRecall

# E
:0
|
```

The first block delivers the email to your INBOX. Pay attention to append the character 'c' also to your recipe so that the incoming email gets cloned before it gets delivered to your mailbox so that the processing of the procmailrc-file is continued.

The second block calls *SpontaneousRecall* to recall the email only if the recall email contains the label RECALL-ME in the subject. Be sure to use the same label that you configured in the file recall.properties.

The last block deletes a remaining email if it wasn't a recall email. This is necessary since procmail expects all emails to be worked up after processing .procmailrc. Since we cloned the email for delivering we may still have one email left if it wasn't handled in the block before.

2.4 SpontaneousRecall on a Remote Machine

Requirements

- Write access to an arbitrary directory
- Installed JRE 5 or higher

Installation and Configuration

Installation is done like on the mailserver. Just copy the files to an arbitrary directory on your machine. Run the tool *at.nolf.Configuration* or adjust the properties connection.properties and recall.properties by hand. See chapter 2.6 and 2.8 for details about the tool and the possibilities of configuration.

SpontaneousRecall can be started to run once by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

Or you may start it to run as a daemon by executing:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall daemon
```

Note that the time interval of the daemon is configured in the file recall.properties.

2.5 Expiry

As mentioned in chapter 1.3, *Expiry* is a program that finds and deletes expired emails. People can provide an expiration date of two different types in their emails that is supported by this program:

- Expiry date in a headerfield:
  The name of the headerfield is *'Expires'* and the value must follow the date format of RFC 2822. An Example for such a headerfield would be: "Expires: Fri, 29 Aug 2008 00:00:00 +0200"

- Expiry date in the subject:
  The expiry date must be given in the following format: "Expires: yyyy/mm/dd" whereby the position in the subject doesn't matter. For example: "Expires: 2008/09/28"

General Configuration

Like the two other programs, *Expiry* needs to know a few things about the mailaccount that it should handle. It shares this information with them and therefore reads all needed configurations from the file connection.properties. Depending on whether *Expiry* runs directly on the mailserver or on a remote machine, the hostnames may slightly differ (ie. localhost, mail2.uibk.ac.at).

Moreover the behaviour of *Expiry* needs to be configured by providing a file named expiry.properties, where you can define — among other things — if an expired email gets deleted or just marked. Such a file with default values will be created when you run this program for the very first time. See chapter 2.8 to see all possible configurations which you can manually change or add.

Requirements

- Write access to an arbitrary directory
- Installed JRE 5 or higher

Running Expiry
Expiry can be run once, or run as a daemon in configurable time intervals by executing one of the following commands in your command line:

```
java -cp EmailReduction.jar at.nolf.Expiry
```

or as a daemon:

```
java -cp EmailReduction.jar at.nolf.Expiry daemon
```

Note that the time interval of the daemon is configured in the file expiry.properties.

Logs and Statistics
Detailed logs of *Expiry* are written to the file recall.log, but it also writes less detailed statistical informations to a separate file expiry_statistics.log to provide a better overview of the number of successful expirations.

See chapter 2.7 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.6 Configuration Tool

To ease the process of configuration I developed a little tool that helps you with this. It's called *at.nolf.Configure* and can be run by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.Configure
```

It asks you some simple questions and writes your answers into appropriate configuration files. If some configuration files already exist, *Configuration* tells you the current value so that you can keep the old value by simply hitting the enter key.

Sample question:

```
Authentication necessary? [ true | false ] (current: true):
Username: (current: csae7569):
```

2.7 Changing the Log Configuration

One can modify the loglevel, maximum size of logfiles, rotations, etc. by providing one's own log4j configuration. Just unzip the file EmailReduction.jar and copy the file log4j.properties to the root directory. Use the file as a template and change desired lines. The default log4j configuration is overwritten by your own configuration by setting a system property containing the path to your file. Example:

```
java -Dlog4j.configuration=file:./mylog4j.properties \
          -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

2.8 Complete Summary of Configuration

Table 2. Configuration with connection.properties (*Vacation*, *SpontaneousRecall* and *Expiry*)

Key	Description	Default
mail.pop3.host	Hostname of pop3 server	mail2.uibk.ac.at
mail.pop3.folder	Pop3 folder on server	INBOX
mail.pop3.auth	Is authentication at pop3 server necessary?	true
mail.transport.protocol	Protocol that is used to send an email	smtp
mail.smtp.host	Hostname of smtp server	localhost
mail.smtp.port	Port for smtp service	25
mail.smtp.from	Address of the sender (your email address)	You@uibk.ac.at
mail.smtp.starttls.enable	Allows communication over an encrypted layer	true
mail.smtp.auth	Is authentication at smtp server necessary?	false
session.debug	Log the process of sending an email very detailed	false
mail.user	Username for mailaccount (for both pop3 and smtp)	Your Username
mail.password	Encrypted password for mailaccount (pop3 and smtp)	
	Has to be set by using the tool *at.nolf.Configure*	Your Password
vacation.timespan	Time interval between sending two vacation messages	604800000
SendVacationMsgInMs	to one and the same recipient (in ms)	(=1 week)

Table 3. Configuration with recall.properties (only for *SpontaneousRecall*)

Key	Description	Default
recall.indicatorRecallMsg	Text that has to be specified at the very beginning of the subject to indicate a recall message	RECALL-ME
recall.markMessageAsRecalled	If true, the email to recall is only marked as recalled and not deleted	true
recall.markMessageAsRecalled.subject	This prefix is added to the subject to indicate that this email has been recalled	RECALLED:
recall.markMessageAsRecalled.header	This header is added to the recalled email	Recalled
recall.recallByMsgId	If true, *SpontaneousRecall* will try to find the initial email to recall by means of the message-id	true
recall.recallByTextCompare	If true, *SpontaneousRecall* will try to find the initial email to recall by comparing the content	true
recall.respectDateLastRun	If true, *SpontaneousRecall* will only search for recall messages that arrived since the last run	true
recall.timespanDaemonRun	Time interval between two runs of *SpontaneousRecall* when it operates in daemon mode (in ms)	30000

Table 4. Configuration with expiry.properties (only for *Expiry*)

Key	Description	Default
expiry.markMessageAsExpired	If true, the expired email is only marked as expired and not deleted	true
expiry.markMessageAsExpired.subject	This prefix is added to the subject to indicate that this email is expired	EXPIRED:
expiry.markMessageAsExpired.header	This header is added to the expired email	Expired
expiry.respectDateLastRun	If true, *Expiry* will only search for expired messages that arrived since the last run	true
expiry.timespanDaemonRun	Time interval between two runs of *Expiry* when it operates in daemon mode (in ms)	30000