Mit einem Perl-Agenten und Jabber über Ebay-Auktionen informieren

E-Baywatcher

Wer in letzter Sekunde in Versteigerungen eingreifen will, lässt sich rechtzeitig von einem Perl-Agenten daran erinnern. Der Agent sucht auf Ebay nach Stichworten und informiert seinen Mandanten sofort per Instant Message, wenn sich eine passende Auktion dem Ende nähert. Michael Schilli

Das Online-Auktionshaus Ebay bietet nach eigenen Angaben allein in Deutschland Zugriff auf mehr als eine Million Artikel. Um so schwerer wird es bei dieser Angebotsdichte, eine bestimmte Auktion zu finden. Das hier vorgestellte Skript »ebaywatch« (siehe Listing 1) sendet in regelmäßigen Abständen definierbare Suchanfragen an den Ebay-Server und wertet die gefundenen Auktionen nach ihrem Abschlussdatum aus.

Nähert sich eine Auktion ihrem Ende, verpackt das Skript eine kurze Beschreibung mit der URL zur Auktion in eine Instant-Messenger-Nachricht (IM) und schickt diese über den Jabber-Server an einen Gaim-Client [5]. Gaim lässt die Nachricht daraufhin dem interessierten Benutzer auf den Bildschirm flattern (siehe Abbildung 1). Ein Mausklick auf die URL genügt dann, um einzuschreiten und mitzusteigern.

Die Datei ».ebaywatchrc« im Homeverzeichnis des Anwenders legt fest, nach welchen Stichworten der Agent auf der Ebay-Website suchen soll. Jede nicht auskommentierte Zeile steht für eine Suchanfrage. Die Zeilen

```
# ~/.ebaywatchrc
dwl 650
nikon
```

definieren zum Beispiel Anfragen nach der D-Link-Netzwerkkarte DWL-650 und nach allen möglichen Produkten der Firma Nikon.

Das Skript durchforstet die Titelfelder der einzelnen Auktionen und unterstützt mit den in [2] beschriebenen Kürzeln sogar erweiterte Suchfunktionen. So findet zum Beispiel die Zeile »foto -nikon« alle Fotoartikel, aber keine Produkte der Firma Nikon. Der Suchstring »beatles (dvd,cd)« fördert alle Angebote von Beatles-CDs und -DVDs ans Licht.

CPAN-Module für Jabber und Ebay

Wie so oft hat sich bereits jemand nach dem Motto „I wrote code, so you don't have to“ (siehe [6]) daran gemacht, die Suche in den Ebay-Beständen als Perl-Modul zu realisieren. In den CPAN-Archiven findet sich Martin Thurns »WWW::Search::Ebay« mit dem Modul »WWW::Search::Ebay::ByEndDate«. Es generiert die vom Benutzer vorgegebenen Suchabfragen und sortiert die eintrudelnden Ergebnisse nach dem Enddatum – perfekt!

Das ebenfalls beim CPAN erhältliche »Net::Jabber«-Modul von Ryan Eatmon enthält ein vollständiges API, um funktionsfähige Jabber-Clients zu schreiben. Ebaywatch nutzt lediglich einen kleinen Teil davon: Es muss sich im Bedarfsfall nur schnell mit dem Jabber-Server »jabber.org« auf Port 5222 verbinden, ihm seine Anwesenheit mitteilen, eine Nachricht an den Mandanten abschicken und sich dann gleich wieder verabschieden. Wer mehr Funktionen nutzen will, findet sie in dem Buch des mächtigen Jabberers DJ Adams (siehe [3]).

Zwei in einem

Um den Ablauf zu vereinfachen, benutzen sowohl das Überwachungsskript als auch der Empfänger den gleichen Jabber-Account. Das ist möglich, da der Jabber-Server es erlaubt, sich mit einem einzigen Benutzernamen von mehreren IM-Clients aus gleichzeitig einzuloggen.

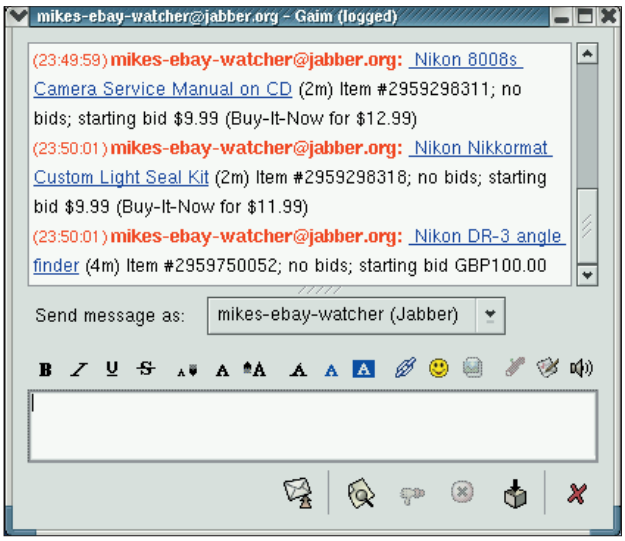

Abbildung 1: Ein Perl-Agent meldet pünktlich drei Ebay-Auktionen, die in wenigen Minuten beendet sein werden. Der Anwender hat den Agenten nach bestimmten Suchbegriffen auf dem Ebay-Server suchen lassen.

Damit der Server die verschiedenen eingeloggten Benutzer voneinander unterscheiden kann, qualifizieren diese sich zusätzlich zu ihrem Benutzernamen noch mit einer so genannten Resource. Sie besteht aus einem String, der jeden Client in Verbindung mit dem Benutzernamen eindeutig identifiziert. Das Überwachungsskript benutzt den Resource-Namen »ebaywatcher«, während Gaim seinen eigenen Resource-String definiert. Neuere Gaim-Versionen erlauben es auch, die Resource über ein Dialogfenster einzustellen.

Eine Sache der Einstellung

Die Konfigurationssektion am Anfang von »ebaywatch« definiert in Zeile 14 (Listing 1) die Variable »$EBAY_HOST«. Sie gibt an, welchen der vielen internationalen Ebay-Server das Skript kontaktieren soll. Im Listing ist »http://search.ebay.de« festgelegt. Wer das amerikanische Original bevorzugt, setzt »$EBAY_HOST« auf »http://search.ebay.com«. Die Variable »$MINS_TO_END« bestimmt, wie viele Minuten vor Ende einer Auktion die Blitznachricht eintreffen soll – voreingestellt sind zehn.

In der Datei aus »$SEEN_DB_FILE« legt Ebaywatch einen persistenten Hash ab. Dort speichert das Programm Statusinformationen, die es nicht nur innerhalb eines Laufs, sondern über mehrere Aufrufe hinweg benötigt. In der Zeile 28 bindet der »tie«-Befehl den globalen Hash »%SEEN« an die konfigurierte Datei. Das Modul »DB_File« sorgt dafür, dass jede Änderung im Hash auch in diesem File landet – vergleichbar einer einfachen Datenbank.

Die Option »O_RDWR« sorgt für Lese- und Schreibrechte, während »O_CREAT« den »tie()« dazu veranlasst, die Datei bei Bedarf neu anzulegen. Zeile 32 sorgt dann dafür, dass der Hash sich auch wieder ordnungsgemäß von der Datei abkoppelt, falls das Programm abbricht.

Programme, die wie Ebaywatch im Hintergrund laufen, geben Statusmeldungen am besten in eine Logdatei aus. Dafür sorgen die drei Funktionen »DEBUG()«, »INFO()« und »LOGDIE()« aus dem Log::Log4perl-Fundus. Im Listing ist »/tmp/ebaywatch.log« als Logfile ausgewählt.

Die Konstruktion des Ebay-Objekts in Zeile 34 ist etwas ungewöhnlich, sie erfolgt über die »new«-Methode der »WWW::Search«-Klasse, die den String »'Ebay::ByEndDate'« als Parameter erhält. Die anschließende While-Schleife ab Zeile 39 iteriert durch die Zeilen der Datei »~/.ebaywatchrc«, eliminiert die Kommentare und Leerzeilen und merkt sich den jeweils geforderten Suchausdruck in »$term«.

Bitte nicht stören

Der persistente Hash »%SEEN« speichert unter den Schlüsseln »"url/$url"« die URLs von Auktionen, die Ebaywatch schon gemeldet hat und daher nicht erneut per IM-Nachricht senden soll. Es kann vorkommen, dass Ebay für einige Suchbegriffe nur Auktionen anzeigt, die so weit in der Zukunft liegen, dass sich eine weitere Anfrage in nächster Zeit nicht lohnen würde. Schließlich soll das Skript die Ebay-Betreiber nicht mit zu vielen Anfragen verärgern, die sowieso keine neuen Informationen enthalten können. Ebaywatch speichert diese Suchbegriffe unter den Schlüsseln »"notuntil/$term"« in »%SEEN« und legt als zugehörigen Wert die lokale Unix-Zeit der nächsten Abfrage fest.

Keine Sonderzeichen

Zeile 58 wandelt Sonderzeichen in URL-kompatible Sequenzen um und Zeile 60 bereitet die gesamte URL der Suchanfrage nach den Ebay-Richtlinien vor. Die While-Schleife ab Zeile 63 holt mit der »next_result()«-Methode die Ergebnisse. Folgende Methoden liefern wichtige Auktions-Informationen:

- »url()«: URL zur Auktion
- »title()«: Kurzbeschreibung
- »description()«: die Auktionsnummer, Anzahl der Gebote, aktuelles Gebot
- »change_date()«: verbleibende Zeit (»2T 02Std 29Min« oder »2d 02h 29m«)

Wegen der unterschiedlichen Zeitanzeige bei »change_date()« zwischen dem amerikanischen und dem deutschen Format wandelt die ab Zeile 99 definierte Funktion »minutes()« einfach beide Darstellungen in Minuten um. Sie nutzt dazu eine simple Mustererkennung mit regulären Ausdrücken.

In der Ergebnisschleife erscheinen die zeitlich nächstgelegenen Auktionen im-

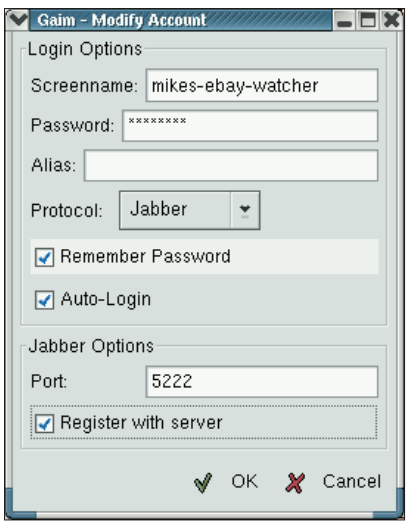

Abbildung 2: Gaim beherrscht die gängigen IM-Protokolle, darunter Jabber. Hier wird der Benutzer »mikes-ebay-watcher« angelegt. Gaim loggt sich dann automatisch beim Server ein.

mer zuerst, das garantiert »WWW::Search::Ebay::ByEndDate«. Wenn also Zeile 77 feststellt, dass die nächste Auktion in der Ergebnisliste mehr als zehn Minuten in der Zukunft liegt und sie die nächste Untersuchung zehn Minuten vor Ende dieser Auktion anberaumt, kann das Skript getrost alle späteren Auktionen außer Acht lassen und mit »last« die Schleife abbrechen. In diesem Fall zieht es zehn Minuten von der Endzeit der Auktion ab, wandelt diesen Wert in die lokale Unix-Uhrzeit in Sekunden um und speichert diese unter »"notuntil/$term"« im permanenten Hash ab.

Steht der Ergebniszähler »$hits« am Ende der Schleife immer noch auf »0«, gab es zum angegebenen Suchbegriff keinen Treffer und die nächste Suche verschiebt sich um einen Tag.

Um eine Jabber-Nachricht abzuschicken, baut Ebaywatch in Zeile 86 den HTML-Code für einen Link und die verfügbaren Auktionsinformationen zusammen und eliminiert in Zeile 90 mit einem regulären Ausdruck alle nicht druckbaren Zeichen. Die Funktion »jabber

Listing 1: Ebay-Auktionen per Jabber überwachen

```
001 #!/usr/bin/perl
002 ##########################################
003 # ebaywatch
004 # Mike Schilli, 2003 (m@perlmeister.com)
005 ##########################################
006 use warnings;
007 use strict;
008
009 our $JABBER_ID      = "mikes-ebay-watcher";
010 our $JABBER_PASSWD  = "********";
011 our $JABBER_SERVER  = "jabber.org";
012 our $JABBER_PORT    = 5222;
013 our $SEEN_DB_FILE   = "/tmp/ebaywatch";
014 our $EBAY_HOST = "http://search.ebay.de";
015 our $MINS_TO_END    = 10;
016 our $RC_FILE   = "$ENV{HOME}/.ebaywatchrc";
017 our %SEEN;
018
019 use Net::Jabber qw(Client);
020 use DB_File;
021 use Log::Log4perl qw(:easy);
022 use WWW::Search::Ebay;
023
024 Log::Log4perl->easy_init(
025   { level => $DEBUG,
026     file => ">>/tmp/ebaywatch.log" });
027
028 tie %SEEN, 'DB_File', $SEEN_DB_FILE,
029     O_CREAT|O_RDWR, 0755 or
030     LOGDIE "tie: $SEEN_DB_FILE ($!)";
031
032 END { untie %SEEN }
033
034 my $search = WWW::Search->new(
035                       'Ebay::ByEndDate');
036 open FILE, "<$RC_FILE" or
037     LOGDIE "Cannot open $RC_FILE";
038
039 while(<FILE>) {
040     # Discard comment and empty lines
041   s/^\s*#.*//;
042   next if /^\s*$/;
043   chomp;
044
045   my $term = $_;
046   my $hits = 0;
047
048   if(exists $SEEN{"notuntil/$term"} and
049      time() < $SEEN{"notuntil/$term"}) {
050     DEBUG "Not checking '$term' until ",
051           scalar localtime
052           $SEEN{"notuntil/$term"};
053     next;
054   }
055
056   DEBUG "Searching for '$term'";
057
058   my $q = WWW::Search::escape_query($term);
059
060   $search->native_query($q,
061       { ebay_search_host => $EBAY_HOST } );
062
063   while (my $r = $search->next_result()) {
064     $hits++;
065     DEBUG "Result: ", $r->url(),
066           " ", $r->title(),
067           " ", $r->description(),
068           " ", $r->change_date();
069
070     if($SEEN{"url/" . $r->url()}) {
071       DEBUG "Already notified";
072       next;
073     }
074
075     my $mins = minutes($r->change_date());
076
077     if($mins > $MINS_TO_END) {
078       $SEEN{"notuntil/$term"} =
079         time + ($mins - $MINS_TO_END) * 60;
080       last;
081     }
082
083     INFO "Notify for ", $r->description;
084     $SEEN{"url/" . $r->url()}++;
085
086     my $msg = "<A HREF=" . $r->url() .
087       ">" . $r->title() .  "</A> " .
088       "(${mins}m) " .  $r->description;
089
090     $msg =~ s/[^[:print:]]//g;
091     jabber_send($msg);
092   }
093     # Pause for 1 day on no results
094   $SEEN{"notuntil/$term"} =
095     time + 24*3600 unless $hits;
096 }
097
098 ##########################################
099 sub minutes {
100 ##########################################
101     my($s) = @_;
102
103     my $min = 0;
104
105     $min += 60*24*$1 if $s =~ /(\d+)[dT]/;
106     $min += 60*$1 if $s =~ /(\d+)[hS]/;
107     $min += $1 if $s =~ /(\d+)[mM]/;
108
109     return $min;
110 }
111
112 ##########################################
113 sub jabber_send {
114 ##########################################
115     my($message) = @_;
116
117     my $c = Net::Jabber::Client->new();
118
119     $c->SetCallBacks(presence => sub {});
120
121     my $status = $c->Connect(
122         hostname => $JABBER_SERVER,
123         port     => $JABBER_PORT,
124     );
125
126     LOGDIE "Can't connect: $!"
127         unless defined $status;
128
129     my @result = $c->AuthSend(
130         username => $JABBER_ID,
131         password => $JABBER_PASSWD,
132         resource => 'ebaywatcher',
133     );
134
135     LOGDIE "Can't log in: $!"
136         unless $result[0] eq "ok";
137
138     $c->PresenceSend();
139
140     my $m = Net::Jabber::Message->new();
141     my $jid = "$JABBER_ID" . '@' .
142               "$JABBER_SERVER/GAIM";
143     $m->SetBody($message);
144     $m->SetTo($jid);
145     DEBUG "Jabber to $jid: $message";
146     my $rc = $c->Send($m, 1);
147
148     $c->Disconnect;
149 }
```

_send()« nimmt dann die String-Nachricht als Parameter entgegen und erstellt ein neues »Net::Jabber::Client«-Objekt. Nach einer erfolgreichen Kontaktaufnahme mit »jabber.org« via »Connect()« sendet der Client seinen Benutzernamen und das Passwort für den Jabber-Account (siehe Kasten „Voraussetzungen für Ebaywatch"). Den »resource«-Parameter setzt das Skript wie oben beschrieben auf »ebaywatcher«.

Präsenz zeigen

In Zeile 138 teilt die Methode »PresenceSend()« dem Jabber-Server mit, dass der Skript-Client anwesend ist. Dieser Client interessiert sich nicht für die Präsenz anderer Clients, daher setzt er in Zeile 119 die korrespondierende Callback-Funktion auf Ignorieren:

```
$c->SetCallBacks(
    presence => sub {}
);
```

Folglich ignoriert das Skript die Presence-Mitteilungen anderer Clients.
Die Jabber-ID des Mandanten setzt sich zusammen aus dem Benutzernamen und dem als »@jabber.org« angehängten Jabber-Server. Die Send-Methode schickt die als »Net::Jabber::Message«-Objekt verkleidete Nachricht an den Server, der sie auch dann entgegennimmt, wenn der Mandant gar nicht online ist. Das angehängte »/GAIM« bestimmt, dass die Methode »SendTo()« die Nachricht nicht an den Jabber-Client im Skript »ebaywatch« schickt (der ja unter der Resource »ebaywatcher« eingeloggt ist), sondern an den unter derselben Benutzer-ID (im Beispiel »mikes-ebay-watcher«) eingeloggten Gaim-Client, der daraufhin automatisch den Resource-Namen »GAIM« definiert.

Auktion frei

Wenn das Skript von der Kommandozeile aus einwandfrei läuft (ein »tail -f *Logdatei*« hilft beim Überwachen), startet folgende Zeile in der Cron-Datei das Skript alle fünf Minuten:

```
*/5 * * * * /home/mschilli/bin/ebaywatch
```

Wer den ganzen Tag lang über einen Instant Messenger kommuniziert, wird sich über ein paar zusätzliche Nachrichten freuen, die der virtuelle Freund schickt. Und dann einfach draufgeklickt und mitgesteigert! *(mwe/fjl)*

Voraussetzungen für Ebaywatch

Wie üblich sind die erforderlichen Zusatzmodule »WWW::Search::Ebay«, »Net::Jabber« und »Log::Log4perl« über eine CPAN-Shell zu installieren:

```
perl -MCPAN -eshell
cpan> install WWW::Search::Ebay
cpan> install Net::Jabber
cpan> install Log::Log4perl
```

Die ersten beiden fordern weitere Module vom CPAN an. Wenn die CPAN-Shell-Option »prerequisites_policy« auf »follow« gesetzt ist, werden diese automatisch installiert.
Wie detailliert Log4perl Ereignisse in der Logdatei protokollieren soll, legt die Option »level« in Zeile 25 fest. Der Defaultwert »$DEBUG« loggt am meisten, »$INFO« nur die wichtigsten Informationen und »$ERROR« gibt nur schwere Fehler aus. Wer nicht will, dass die Logdatei zu lang wird, erweitert die »Log::Log4perl«-Konfiguration mit einem »RollingFileAppender«, der Dateien nur bis zu einer vordefinierten Größe vollschreibt, eine einstellbare maximale Zahl von Dateien anlegt und die erste wieder überschreibt, wenn die Maximalzahl erreicht ist (siehe [4]).

Jabber-Geschnatter

Einen Jabber-Account legt man am einfachsten mit Gaim an, einem anpassungsfähigen IM-Client, der alle gängigen Instant-Messenger-Protokolle spricht. Unter [5] gibt es das Programm zum Download. Bei älteren Versionen von Gaim muss der Anwender das Jabber-Plugin manuell nachladen, indem er unter dem Menüpunkt »Tools | Plugins« die »jabberlib.so« auswählt und hinzulädt (siehe Abbildung 3). Ein Klick auf »Add« unter »Tools | Accounts« öffnet ein Fenster mit dem Formular aus Abbildung 2. Gaim merkt sich das Passwort und loggt sich nach dem Start automatisch am Jabber-Server an.

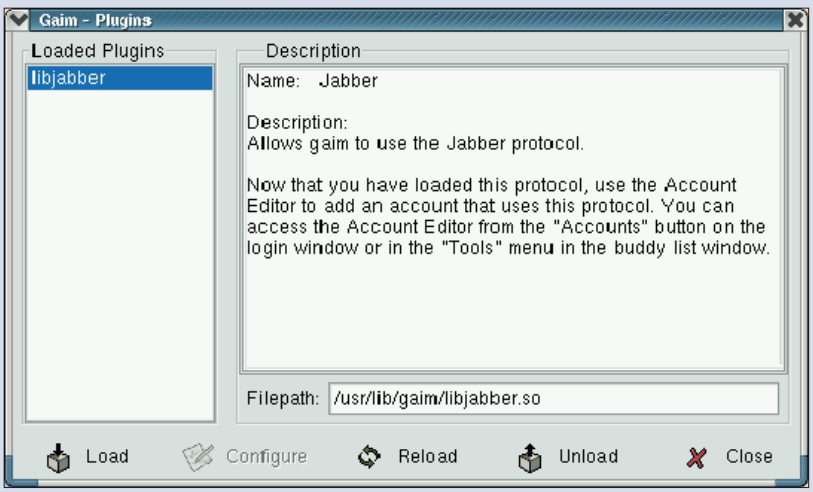

Abbildung 3: Der Ebay-Agent nutzt das Jabber-Protokoll für seine Benachrichtigungen. In älteren Versionen von Gaim muss der Anwender dazu allerdings ein Plugin manuell nachladen.

Infos

[1] Listings zu diesem Artikel: [ftp://ftp.linux-magazin.de/pub/listings/magazin/2004/01/Perl/]
[2] David A. Karp, „eBay Hacks: 100 Industrial-Strength Tips and Tools": O'Reilly 2003, ISBN 0-59600-564-4
[3] DJ Adams, „Programming Jabber": O'Reilly 2002, ISBN 0-59600-202-5
[4] Logdateien automatisch begrenzen und rotieren: [http://log4perl.sourceforge.net/releases/Log-Log4perl/docs/html/Log/Log4perl/FAQ.html#how_can_i_roll_over_my_logfiles_automatically_at_midnight]
[5] Gaim, ein Instant-Messenging-Client: [http://gaim.sourceforge.net]
[6] T-Shirt „I wrote code so you don't have to": [http://www.thinkgeek.com/interests/oreilly/tshirts/6067]

Der Autor

Michael Schilli arbeitet als Web-Engineer für AOL/Netscape in Mountain View, Kalifornien. Er hat „Goto Perl 5" (deutsch) und „Perl Power" (englisch) für Addison-Wesley geschrieben und ist unter [mschilli@perlmeister.com] zu erreichen. Seine Homepage ist [http://perlmeister.com].