Monitoring Ebay Auctions with a Perl Agent and Jabber

E-Baywatcher

If you are looking to snipe an Ebay auction at the last minute, why not use a Perl agent that performs a keyword search of Ebay and notifies you of the imminent end of auction by instant message. Thus, allowing you to place a bid at the end and avoid being outbid on that precious antique computer part you have always hankered for.

BY MICHAEL SCHILLI

Ebay, the self-acclaimed World Online Marketplace, has around 16 million items on auction in the US on any given day. Needless to say, it can be difficult to find a specific auction considering the sheer bulk of items listed.

The script we will be looking at this month, *ebaywatch* (see Listing 1), regularly transmits search requests to the Ebay server and evaluates the auctions, returned by the search, by their ending date. When an auction is drawing to a close, the script wraps a short description and the URL of the auction in an instant message, and uses the Jabber server to bundle this off to a *gaim* client [5]. Gaim then pops the message up on the requesting user's display (see Figure 1), where a mouse click on the URL will take the user right to the auction, in time to bid.

The *.ebaywatchrc* file in the user's home directory tells the agent what keywords to search the Ebay website with. Any line without a hash sign represents a search request. For example, the following lines

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

define a request for a D-Link "DWL-650" network interface card, and any products by Nikon. The script searches the title fields of online auctions, and can even support advanced searching features using the abbreviations described in [2]. Thus, for example, a line such as *photo -nikon* will retrieve any photo articles except Nikon products, and *"beatles (dvd,cd)"* will retrieve any Beatles CDs and DVDs.

CPAN Modules

As is often the case, someone has been there before (true to the premise "I wrote code so you don't have to" – see [6]), so there is a Perl module that performs Ebay searches. Look for the *WWW::Search::Ebay* distribution by Martin Thurn, including the *WWW::Search::Ebay::ByEndDate* module, at CPAN. This provides all the functionality you need to submit search requests, and sort the results the search returns by ending date – just what the doctor ordered!

Another module available from CPAN, *Net::Jabber* by Ryan Eatmon, provides a complete API that allows you to write a functional Jabber client. *ebaywatch* only uses a small part of its potential. When needed, the client simply has to connect to the Jabber server *jabber.org* on port 5222, introduce itself to the server, send a message to the user, and then finally say goodbye. If you need more functionality than this, try out the book by the mighty Jabberer, DJ Adams (see [3]).

To simplify the procedure, both the monitoring script and the receiver use the same Jabber account. This is possible because the Jabber server allows you to log in simultaneously from multiple IM clients using the same username.

To allow the server to distinguish between multiple logins with the same username, it adds a so-called resource. This is a string that uniquely identifies each client in combination with the username. The monitoring script uses "ebaywatcher" as its resource name, whereas Gaim defines a resource string of its own.

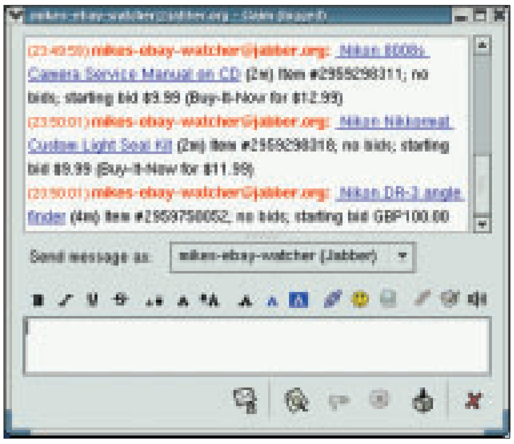
A Question of Preferences

The configuration section of *ebaywatch* defines the variable *$EBAY_HOST* in line 16 of the script. This tells the script which of the many international Ebay servers it should contact. The Listing assumes *http://search.ebay.com*. If you prefer to use the UK search server, simply set *$EBAY_HOST* to *http://search.ebay.co.uk*. *$MINS_TO_END* tells the script how many minutes notice you require before the end of the auction – the default is 10.

ebaywatch uses the file *$SEEN_DB_FILE* to store a persistent hash and thus save status information between calls. In line 28, the *tie* command uses the *DB_File* module to bind the global hash, *%SEEN*, to the configured file. The *O_RDWR* option sets read and write privileges, and *O_CREAT* tells *tie()* to create the file if it does not already exist. Line 32 ensures that the hash unbinds from the file, if the program ends.

For programs that run in the background, like *ebaywatch*, a logfile is the best place for status messages. And this is where the functions *DEBUG()*, *INFO()*, and *LOGDIE()* from Log::Log4perl put them. Our Listing uses */tmp/ebaywatch.log* as its logfile.

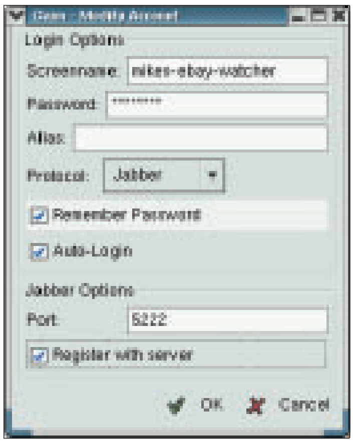
The construction of the Ebay object in line 34 is slightly unusual; it uses the *WWW::Search* class' *new* method, which expects to be passed a *'Ebay::ByEndDate'* string as a parameter. The while loop in line 39 onwards, iterates through the lines of the *~/.ebaywatchrc* file,



(23:49:59) mikes-ebay-watcher@jabber.org: Nikon 8008s Camera Service Manual on CD (2m) Item #2958298311; no bids; starting bid $9.99 (Buy-It-Now for $12.99)
(23:50:01) mikes-ebay-watcher@jabber.org: Nikon Nikkormat Custom Light Seal Kit (2m) Item #2958298318; no bids; starting bid $9.99 (Buy-It-Now for $11.99)
(23:50:01) mikes-ebay-watcher@jabber.org: Nikon DR-3 angle finder (4m) Item #2959750052; no bids; starting bid GBP100.00

Send message as: mikes-ebay-watcher (Jabber)

Figure 1: A Perl agent reporting three Ebay auctions that will end in a few minutes. The user told the agent to search the Ebay server for specific keywords



Login Options
Screenname: mikes-ebay-watcher
Password: ********
Alias:
Protocol: Jabber
[✓] Remember Password
[✓] Auto-Login
Jabber Options
Port: 5222
[✓] Register with server

Figure 2: The Gaim Instant Messenger creating a user called "mikes-ebay-watcher".

eliminating comments and empty lines, and placing the search key requested in each line in *$term*.

Do Not Disturb!

The persistent hash, *%SEEN*, stores the URLs of auctions where *ebaywatch* has already alerted the user, in the *"url/$url"* keys; it will not generate another instant message for these auctions.

Some searches may return Ebay auctions which are so far in the future that it would not make sense to repeat the search in the near future. After all, the idea of the script is not to annoy the powers that be at Ebay with a bunch of

ebaywatch Requirements

As usual, you will need to install the additional modules *WWW::Search::Ebay*, *Net::Jabber*, and *Log::Log4perl* using the CPAN Shell:

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

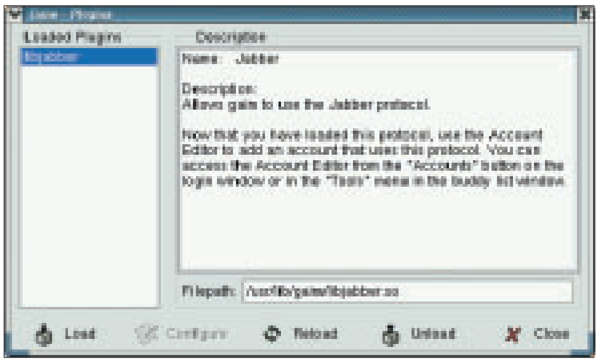
The first two lines request additional modules from CPAN. If the CPAN Shell *prerequisites_policy* option is set to *follow*, these modules will be installed automatically.

The *level* option in line 25 specifies the logging detail level for Log4perl. The default, *$DEBUG*, will create the most entries, *$INFO* logs only the most important information, and *$ERROR* logs only critical errors. To avoid bloating the logfile, you might like to add a *RollingFileAppender* to your *Log::Log4perl* configuration; this allows logfiles to grow to a pre-defined limit, to create a pre-definable maximum number of files, and start overwriting when this threshold has been reached (see [4]).

Jabber Talk

The easiest way to create a Jabber account is to use Gaim, a flexible IM client that can speak all the major Instant Messenger protocols. The program can be downloaded from [5]. You need to install the Jabber plug-in manually if you have an older version of Gaim; select *jabberlib.so* below *Tools | Plugins* to load the plug-in (see Figure 3).

Clicking on *Add* below *Tools | Accounts* opens a window with a form, as shown in Figure 2. Gaim remembers your password and automatically logs on to the Jabber server when launched.



Loaded Plugins
libjabber

Description
Name: Jabber

Description:
Allows gaim to use the Jabber protocol.

Now that you have loaded this protocol, use the Account Editor to add an account that uses this protocol. You can access the Account Editor from the "Accounts" button on the login window or in the "Tools" menu in the buddy list window.

Filepath: /usr/lib/gaim/libjabber.so

Figure 3: The Ebay agent uses the Jabber protocol for messaging. Older versions of Gaim need to install a plug-in

searches that will not return new results anyway. *ebaywatch* stores this kind of search key in the *"notuntil/$term"* key of *%SEEN* adding the value for the local Unix time of the next search request.

No special characters

Line 58 converts special characters to URL compatible sequences, and line 60 organizes the whole URL of the search request to reflect Ebay syntax. The while loop in line 63 onwards, uses the *next_result()* method to get the results. The following methods return auction-critical information:

- *url()*: URL of the auction
- *title()*: short description
- *description()*: auction number, number of bids, current bid
- *change_date()*: remaining time (*2D 02H 29M*)

WWW::Search::Ebay::ByEndDate ensures that the next auction to close will appear at the top of the result loop. Thus, if line 77 discovers that the next auction in the list ends after more than 10 minutes in the future, it will schedule the next search ten minutes before the end of the auction and ignore any auctions later than this one, and use *last* to exit the loop. Here, the script subtracts 10 minutes from the ending time of the auction, converts this value to local Unix time in seconds, and saves the result in *"notuntil/$term"* in the permanent hash.

If the hit counter *$hits* still has a value of *0* at the end of the loop, there are no results for the current search key, and the next search is postponed by one day.

To send a Jabber message, *ebaywatch* collates the HTML code for a link with any available auction data in line 86, using a regex to eliminate any non-printable characters in line 90. The *jabber_send()* function then accepts the message string as a parameter and creates a new *Net::Jabber::Client* object. After issuing a *Connect()* to contact *jabber.org*, the client sends its username and the Jabber account password (see insert "*ebaywatch* Requirements"). The script sets the *resource* parameter to *ebaywatcher* as described previously. Line 138 tells the Jabber server that the script client is alive; the callback function for other clients is set to *ignore* in line 119 with:

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

A Jabber ID comprises of the username and the Jabber server attached as

Listing 1: Using Jabber to Monitor Ebay Auctions

```
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.com";
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_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
```

@jabber.org. The send method then sends the message, which is wrapped in a *Net::Jabber::Message* object on to the server. The server will accept the message even if the user is not currently online. The */GAIM* suffix indicates that *SendTo()* is not sending the message to the Jabber client in the script, *ebaywatch* (which is currently logged on as the *ebaywatcher* resource), but to the running Gaim client, which is logged on as the same user ID (this is the *mikes-ebaywatcher* name in our example, see Figure 2), and automatically assumes the resource name *GAIM*.

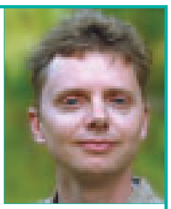
Timing

After making sure that the script runs perfectly in the command line, (using the command *tail -f* logfile will help you check this), use the following crontab entry to launch the script with a five minute frequency:

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

If you have the instant messenger running, you can look forward to a few messages from your new "virtual friend". Then it's up to you to click and bid! ■

THE AUTHOR



Michael Schilli works as a Web engineer for AOL/Netscape in Mountain View, California. He wrote "Perl Power" for Addison-Wesley and can be contacted at mschilli@perlmeister.com. His homepage is at http://perlmeister.com.

INFO

[1] Listings for this article: *ftp://www.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] Automatically restricting growth of and rotating logfiles: *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 homepage: *http://gaim.sourceforge.net*

[6] T-Shirt "I wrote code so you don't have to": *http://www.thinkgeek.com/interests/oreilly/tshirts/6067*

Listing 1: Using Jabber to Monitor Ebay Auctions

```
    if($mins > $MINS_TO_END) {
      $SEEN{"notuntil/$term"} =
        time + ($mins - $MINS_TO_END) * 60;
      last;
    }

    INFO "Notify for ", $r->description;
    $SEEN{"url/" . $r->url()}++;

    my $msg = "<A HREF=" . $r->url() .
      ">" . $r->title() . "</A> " .
      "(${mins}m) " . $r->description;

    $msg =~ s/[^[:print:]]//g;
    jabber_send($msg);
  }
      # Pause for 1 day on no results
  $SEEN{"notuntil/$term"} =
     time + 24*3600 unless $hits;
}

################################
sub minutes {
################################
    my($s) = @_;

    my $min = 0;

    $min += 60*24*$1 if $s =~ /(\d+)[dD]/;
    $min += 60*$1 if $s =~ /(\d+)[hH]/;
    $min += $1 if $s =~ /(\d+)[mM]/;

    return $min;
}

################################
sub jabber_send {
################################
    my($message) = @_;

    my $c = Net::Jabber::Client->new();

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

    my $status = $c->Connect(
        hostname => $JABBER_SERVER,
        port     => $JABBER_PORT,
    );

    LOGDIE "Can't connect: $!"
        unless defined $status;

    my @result = $c->AuthSend(
        username => $JABBER_ID,
        password => $JABBER_PASSWD,
        resource => 'ebaywatcher',
    );

    LOGDIE "Can't log in: $!"
        unless $result[0] eq "ok";

    $c->PresenceSend();

    my $m = Net::Jabber::Message->new();
    my $jid = "$JABBER_ID" . '@' .
              "$JABBER_SERVER/GAIM";
    $m->SetBody($message);
    $m->SetTo($jid);
    DEBUG "Jabber to $jid: $message";
    my $rc = $c->Send($m, 1);

    $c->Disconnect();
}
```