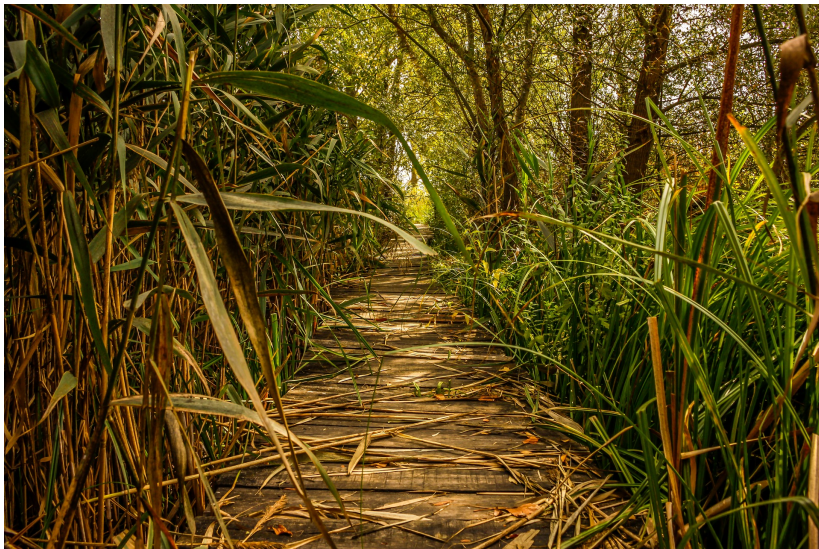


Email with Perl

Stefan Hornburg (Racke)

French Perl Workshop 2019, Strasbourg, 22nd June

Email with Perl



Overview

- Email structure
- Sending emails
- IMAP
- Parsing emails
- Use Cases

Email structure

- Header
- Body

Email structure

- Attachments
- HTML / Plaintext

Email structure

```
Email::MIME->new( $message )->debug_structure;
```

Email structure

```
+ multipart/mixed;  
  + application/octet-stream; name=83AKE9.pdf  
  + multipart/alternative;  
    + text/plain; charset=UTF-8  
    + text/html; charset=UTF-8
```

Sending emails

Too many ways to do it ...



Module (Legacy)

- Net::SMTP
- MIME::Lite

Module

Send:

- Email::Send (deprecated)
- Email::Sender
- Email::Sender::Simple
- Email::Stuffer

Misc:

- Email::MIME
- MIME::Entity
- Mail::Box

Email::Stuffer example

```
Email::Stuffer->from('"Racke" <racke@linuxia.de>')  
->to('"Sympa " <racke@racke.pm>')  
->subject('Changes in Module Foo::Bar')  
->text_body("Baz.\n")  
->send;
```

Benefits of Email::Stuffer

- Syntax Sugar
- Encoding
- Transports

Example

```
my $email = Email::Simple->create(
    header => [
        From      => '"Module Updates" <info@cpan.pm>',
        To        => '"Racke" <racke@racke.pm>',
        Subject   => 'Changes in Module Foo::Bar',
    ],
    body => "Baz.\n",
);

sendmail($email);
```

Example with attachment

```
my $email = MIME::Entity->build(
    From    => '"Module Updates" <info@cpan.pm>',
    To      => '"Racke" <racke@racke.pm>',
    Subject => 'Changes in Module Foo::Bar',
    Data   => ["Baz.\n"],
);

$email->attach(
    Path => '../perlmail-de-handout.pdf',
    Type => 'application/pdf',
);


sendmail($email);
```


Example with umlaut

```
my $email = Email::Simple->create(
    header => [
        From      => '"Module Updates" <info@cpan.pm>',
        To        => '"Racke" <racke@racke.pm>',
        Subject   => 'Änderungen im Modul Foo::Bar',
    ],
    body => "Baz.\n",
);

sendmail($email);
```

Mail Clients

 Thunderbird

 K9 (Android)

Correction: Example with umlaut

```
my $email = Email::Simple->create(
    header => [
        From      => '"Module Updates" <info@cpan.pm>',
        To        => '"Racke" <racke@racke.pm>',
        Subject   => encode('MIME-Header',
                            'Änderungen im Modul Foo',
                            ),
    ],
    body => "Bar.\n",
);

sendmail($email);
```

Encoded subject

```
To: "Racke" <racke@racke.pm>  
From: "Module Updates" <info@cpan.pm>  
Subject: =?UTF-8?B?w4RuZGVydW5nZW4gaW4gTW9kdWxlIEZvbw==?=
```

Encoding email body

```
my $email = Email::Simple->create(
    header => [
        From      => '"Module Updates" <info@cpan.pm>',
        To        => '"Racke" <racke@racke.pm>',
        Subject   => encode('MIME-Header',
                           'Änderungen im Modul Foo::Bar',
                           ),
    ],
    body => encode('UTF-8', "Überflüssig"),
);

sendmail($email);
```

Transports



Transports

- Sendmail
- SMTP
- Maildir
- Redirect

Sendmail Transport

```
sendmail( $email );
```

Prerequisites for SMTP authentication

- MIME::Base64
- Authen::SASL

SMTP Transport

```
my $transport = Email::Sender::Transport::SMTP->new(  
    host => 'mail.perl.dance',  
    ssl => 1,  
    sasl_username => 'racke@perl.dance',  
    sasl_password => 'nevairbe',  
);  
  
sendmail( $email, { transport => $transport } );
```


Maildir Transport

```
my $maildir = path(File::HomeDir->my_home)
    ->child('Maildir');

my $transport = Email::Sender::Transport::Maildir->new(
    dir => $maildir
);

sendmail( $email , { transport => $transport, } );
```

Redirect Transport

```
$transport_orig = Email::Sender::Transport::Sendmail->new;

$transport = Email::Sender::Transport::Redirect->new({
    transport => $transport_orig,
    redirect_address => 'racke@linuxia.de',
});

sendmail( $email , { transport => $transport, } );
```

Redirect Transport

```
To: racke@linuxia.de
Cc: racke@linuxia.de
From: "Module Updates" <info@cpan.pm>
Subject: Hello world!
Date: Tue, 3 Apr 2018 08:45:59 +0200
X-Intercepted-To: "Racke" <racke@racke.pm>
X-Intercepted-Cc: "Info" <info@racke.pm>
X-Email-Sender-From: info@cpan.pm
X-Email-Sender-To: racke@linuxia.de
X-Email-Sender-To: racke@linuxia.de
Lines: 1
```

Here we go.

Embedding images

- Images as links
- Base64 encoded
- CID Inline

Emails from templates

```
my $html = template $template, $tokens,  
    { layout => 'email' };  
  
my $f      = HTML::FormatText::WithLinks->new;  
my $text   = $f->parse($html);
```

Emails from templates

```
email {
  %args,
  body  => encode( 'UTF-8', $text ),
  type  => 'text',
  attach => {
    Charset => 'utf-8',
    Data    => encode( 'UTF-8', $html ),
    Encoding => "quoted-printable",
    Type    => "text/html"
  },
  multipart => 'alternative',
};
```

IMAP

- Modules
- Login
- Select folder
- Search emails
- Download emails

IMAP Modules

- Mail::IMAPClient
- Net::IMAP::Simple
- Net::IMAP::Client

Differences between IMAP modules

- Constructor
 - server
 - ssl / use_ssl
- Supported methods
- Return values
 - select
 - search

Login

```
my $imap = Net::IMAP::Client->new(  
    server => 'mail.linuxia.pm',  
    ssl => 1,  
    port => 993,  
    user => 'racke@racke.pm',  
    pass => 'nevairbe',  
);  
  
$imap->login;
```

Select folder

Select folder (Slash separator):

```
$imap->select('INBOX/Consulting/Bahn');
```

Select folder (Dot separator):

```
$imap->select('INBOX.Consulting.Bahn');
```

Folder separator

```
$imap->separator;  
$imap->select(join($imap->separator,  
qw/INBOX Consulting Bahn/));
```

Search emails

All emails in selected folder:

```
$ids = $imap->search( 'ALL' , 'DATE' );
```

Search emails in selected folder:

```
$ids = $imap->search( { subject => 'Perl' }, 'DATE' );
```

Folders

List of folders:

```
$imap->folders
```

```
[  
  'INBOX/Sympa',  
  'INBOX',  
  'Sent',  
  'Drafts',  
  'Trash'  
]
```

Headers

```
my $summary = $imap->get_summaries($ids);
```

Download emails

Download complete email:

```
my $message = $imap->get_rfc822_body($id);
```


Delete emails

Set deleted flag:

```
$imap->delete_message($id);
```

Delete emails

Remove permanently:

```
$imap->expunge()
```

Delete emails

Move to trash:

```
$imap->move('INBOX.Trash', $id);
```

Modules for parsing emails

- Email::MIME
- Email::Simple
- MIME::Parser

Read email

- Read email from IMAP:

```
my $message = $imap->get_rfc822_body($id);
```

- Read email from file:

```
use Path::Tiny;
```

```
my $message = path('email.eml')->slurp;
```

Email::MIME

- Parse email:

```
my $parser = Email::MIME->new($message);
```

Email::MIME

Subject header:

```
print $parser->header_str('Subject');
```

Umlaute with Perl: `ü` `ä` `ö`

Email::MIME

Subject header:

```
print $parser->header_raw('Subject');
```

```
=?UTF-8?B?VW1sYXVOZSB3aXRoIFB1cmw6IM08IM0kIM02?=
```


Email::MIME

Walk MIME parts:

```
my @out;
```

```
$parser->walk_parts(sub {  
    my ($part) = @_;  
    return if $part->subparts;  
  
    if ($part->content_type =~ m{text/html}i) {  
        push @out, $part->body_str;  
    }  
});
```

Extract train ticket

```
$ids = $imap->search('FROM buchungsbestaetigung@bahn.de');

for my $imap_id (@$ids) {
    my $message = $imap->get_rfc822_body($imap_id);
    my $parser = Email::MIME->new($message);
    my @out;

    ...
}
```

Extract train ticket

```
$parser->walk_parts(  
    sub {  
        my ($part) = @_  
        return if $part->subparts;  
  
        if ($part->content_type =~ m{application/octet-stream}i){  
            push @out, [ $part->filename, $part->body ];  
        }  
    }  
);  
  
$out[0][1] > io($out[0][0]);
```

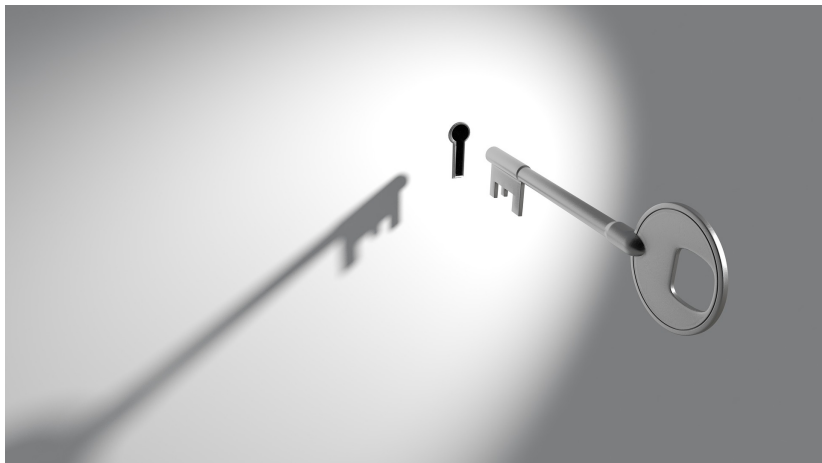
MIME::Parser

```
my $parser = MIME::Parser->new;

# avoid leaving files around.
$parser->output_to_core(1);

my $parsed_message = $parser->parse_data($message);
```

Encryption



Encryption methods

- PGP
- S/MIME

Mail::GnuPG

```
my $gnupg = Mail::GnuPG->new;
```

Mail::GnuPG

Arguments:

- key
- passphrase
- use_agent

Encrypt emails

```
my $gnupg = Mail::GnuPG->new;

# Encrypt
$gnupg->mime_encrypt($message, 'racke@linuxia.de');

# Encrypt and sign
$gnupg->mime_signencrypt($message, 'racke@linuxia.de');
```

Decrypt emails

```
# MIME::Entity object required
```

```
$gnupg->decrypt($entity);
```

```
# Results
```

```
$gnupg->{plaintext}
```

```
$gnupg->{decrypted}
```

Use Cases

- `Dancer2::Plugin::Email (Email::Sender)`
- `Helpdesk::Integration`
- `Sympa`
- `Spamassassin`

Helpdesk::Integration

- Request Tracker
 - Web-Interface
 - Email-Interface

Helpdesk::Integration

- Search IMAP folder
- Parse emails
- Create/update ticket through REST API

Questions



The end

