CPAN特選Perlモジュールで

1歩進んだサーバープログラミング

第4回 Text：情報技研

CGIからのメール送信を簡単に実現するMail::Senderモジュール

CPAN(Comprehensive Perl Archive Network)は、Perlプログラムとドキュメントの膨大なコレクションである。プログラミングに便利な機能を寄せ集めたもので、さしずめ知恵の宝庫といえよう。
本連載では、CPANで4,000個以上提供されているPerlモジュールから便利なものを厳選し、その機能を自分のPerlプログラムから使い、効率よくプログラミングする方法を解説する。

CPAN Powered by Sun
http://www.cpan.org/

CGIからのメール送信

今回は、ウェブの次によく使うメールの送信を行う機能を実現しよう。メールの送信を行うモジュールはいくつかあるが、簡単にいろいろなメールを送信できるMail::Senderモジュールを使ってメールを送信するプログラミング例を解説していく。

Mail::SenderモジュールはPerl標準で利用できるモジュールではないため、まず最初にインストールが必要である。また、Mail::Senderモジュールが内部で利用するMIME::Base64モジュールもインストールしておく必要がある。

Mail::Senderを使った単純なメール送信サンプル

まずはMail::Senderの機能を理解するために、添付ファイル付きメールを送信するサンプルプログラムを見てみよう(コード1)。

実際にこのサンプルコードを実行する際には、宛て先(toの部分)と送信元(fromの部分)のメールアドレスを変更し、smtpの部分でSendmailなどのSMTPサーバーのアドレスを指定する必要がある。このサンプルコードを実行するホストでSMTPサーバーが動作している場合はsmtpは「localhost」のままでよい。fileには添付して送信するファイルを指定しておく。

また、サンプルコードの「$Mail::Sender::Error」変数チェックをしているif文の部分では、メールの送信時にエラーがある場合にエラーの内容を表示するためのものである。うまくメールが送信されないときはエラーが表示されるので、それをチェックしよう。

実際にサンプルコードmail-sender.plで送信されたメッセージの内容をヘッダー部分も含めて見てみる(図1)。見づらくなるため「Received:」ヘッダーは省略してある。メールのヘッダー部分を確認すると「X-Mailer」というヘッダーオプションにMail::Senderで送信された旨が書かれていることがわかる。

Mail::Senderモジュールを使わなければ、これらのヘッダーをいちいち自分で生成しなければならない。添付ファイルがなければこれよりも簡単な内容にはなるが、それでも処理するコードの分量はそれなりになるだろう。ヘッダーの生成からファイルのエンコーディングまで、これだけの処理を手軽に行えてしまうMail::Senderモジュールがいかに便利かがわかるだろう。

コード1　メール送信のサンプルコード(mail-sender.pl)

```
#!/usr/bin/perl
use Mail::Sender;

$send = new Mail::Sender;

$send->MailFile({
  smtp    => 'localhost',
  to      => 'yourname@yourdomain',
  subject => 'Test Mail (with File)',
  from    => 'myname@mydomain',
  msg     => "This is the test mail with file.¥n".
             "It's send by Mail::Sender Perl module.¥n",
  file    => 'filename.zip',
  });

if ($Mail::Sender::Error){
  print $Mail::Sender::Error,"¥n";
  }
```

```
Return-Path: myname@mydomain
Received:  (省略)
To: youname@yourdomain
From: myname@mydomain
Subject: Test Mail
Date: Tue, 8 Apr 2003 11:36:23 +0900
X-Mailer: Perl script "mail-sender.pl"
        using Mail::Sender 0.8.00 by Jenda Krynicky
        running on localhost (127.0.0.1)
        under account "myname"
Message-ID: <20030408_023623_082562.myname@mydomain>
MIME-Version: 1.0
Content-type: multipart/mixed;
        boundary="Message-Boundary-by-Mail-Sender-1049769383"

This message is in MIME format. Since your mail reader does not understand
this format, some or all of this message may not be legible.

--Message-Boundary-by-Mail-Sender-1049769383
Content-type: text/plain; charset=US-ASCII
Content-description: Mail message body
Content-transfer-encoding: 7BIT
Content-disposition: inline

This is the test mail.
It's send by Mail::Sender Perl module.

--Message-Boundary-by-Mail-Sender-1049769383
Content-type: application/octet-stream; name="filename.zip"; type=Unknown;
Content-description: filename.zip
Content-transfer-encoding: BASE64
Content-disposition: attachment

JVBERi0xLjIgDSXi48/TDQogDTggMCBvYmoNPDwNL0xlbmd0aCA5IDAgUg0vRmlsdGVyIC9GbGF0
ZURIY29kZSANPj4Nc3RyZWFtDQpliY2Xy45cJxCGn6DfgeVkkTbFHWkUqS+n12WRRdxL76w4iuWR
FW/8+jlAwaGn61BopNFcvv5O8ON1AASsXz/+EQclwzEIb+LRCJBHJZRcf/rxt/hygAad75Wzav2v
```

図1　mail-sender.plで送信されたメールの内容

実用的なサンプル：メール送信フォームの表示

次に、もう少し具体的なサンプルプログラムを実行してみよう。ここでは、CGIフォームからメールを送信するということを行ってみる。もちろんメールにファイルを添付して送ることも可能で、添付するファイルもCGIでアップロードできるようにしてあるので、ブラウザーだけでメールを送信できる。

このサンプルプログラムは、これまでのようにメールの内容を入力するフォーム画面をCGIモジュールを利用して表示するスクリプトと、フォームに入力されたデータを処理して実際にメールを送信するスクリプトの2つに分かれる。

まず、メールの内容を入力するページ用のサンプルコード（コード2）とその実行例（図2）を示す。

これについてはこれまでと同様にCGIモジュールを利用してフォームを表示しているだけなので特に説明の必要もないだろう。新しいこととしては、添付ファイルの指定のためにCGIアップロード機能を利用している点があるくらいである。

コード2　メール入力フォーム表示のサンプルコード（index.cgi）

```
#!/usr/bin/perl
use CGI qw(:standard);
use CGI::Pretty;

##  入力フォームの表示
print
  header(-charset=>'x-euc-jp'), start_html,
  h2('フォームからメールを送ってみよう'),
  start_multipart_form(-action=>'mail-sender.cgi'),
  table({border=>0},
    Tr([
      td(['To:', textfield('to', '', 50),
          '←宛て先のアドレス(「,」で区切り複数可)']),
      td(['Cc:', textfield('cc', '', 50),
          '←同報送信のアドレス(「,」で区切り複数可)']),
      td(['Subject:', textfield('subject', '', 50),
          '←件名（日本語可）']),
      td(['From:', textfield('from', '', 50),
          '←送信者のアドレス']),
      td(['自分の名前:', textfield('name', '', 50),
          '←送信者の名前（日本語可）']),
      td(['添付ファイル:', filefield('file', '', 40)]),
      ]),
    ),
  br('メールの本文:'), textarea('Message', '', 12, 70), p,
  submit('メール送信'), reset('入力項目リセット'),
  end_form, end_html;
```

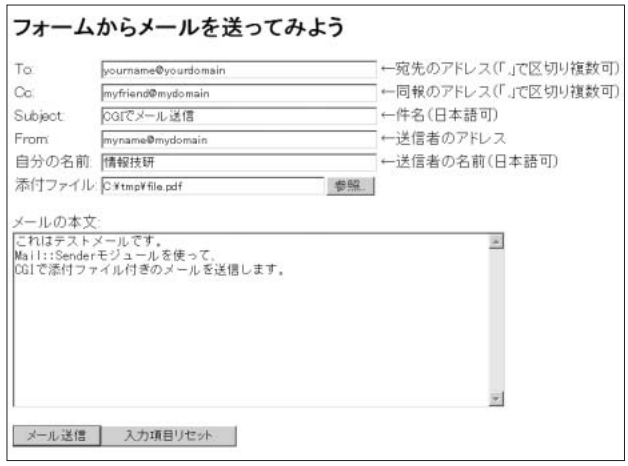

図2　index.cgiの実行結果

実用的なサンプル：メールの送信処理

次に、フォームのデータを受け取り、実際にメールを送信するサンプルコードを示す（コード3）。最初に示したコード1で扱ったメールでは、メールの内容にアルファベットのascii文字しか使っていなかったので単純な処理でよかったが、ここではより実用的なものとするために日本語の処理を行っているため、コードが若干複雑になっている。

図2で示した入力でこのサンプルコードを実行させると図3のような内容のメールが送信される。

メソッドの詳しい解説は後でするが、大まかな処理の流れを説明すると、次のようになる。

・添付ファイルがない場合

MailMsgメソッドでメールを送信する。

・添付ファイルがある場合

OpenMultipart→Body→Send→Attach→Closeの一連のメソッドを組み合わせて使ってメールを送信する。

添付ファイルがある場合でもMailFileメソッドを使わずに複数のメソッドを組み合わせる方法をとっているのは、本文に日本語を使えるようにするためである。

メールで日本語を扱うための処理

それでは、もう少し詳しく説明していく。メールで日本語を使う場合には、送信する際に日本語を7bitの文字列の形式に変換することが望まれる。そのため、ヘッダー部分である件名（Subject）やアドレス（From、To、CC、BCCなど）に含まれる日本語はMIMEエンコードし、本文は7bitのJISコードに変換する作業が必要となる。これを行うためにJcodeモジュールが活躍している。

コード3　メール送信CGIのサンプルコード（mail-sender.cgi）

```
#!/usr/bin/perl

use CGI qw(:standard);
use CGI::Pretty;
use Mail::Sender;
use Jcode;
use File::Path;
use File::Copy;

##  フォーム変数の処理
$to       = param('to');
$cc       = param('cc');
$subject  = jcode(param('subject'))->mime_encode();
$fromAddr = param('from');
$fromName = jcode(param('name'))->mime_encode();
$from     = "$fromName <$fromAddr>";
$message  = jcode(param('Message'))->jis;
$file     = param('file');

##  ファイルの処理
open (FILE, ">$file");
copy ($file, ¥*FILE);

##  メールの送信
$send = new Mail::Sender;
if ($file){
  $send->OpenMultipart({
    smtp => 'localhost', to => $to, cc => $cc,
    from => $from, subject => $subject});
  $send->Body('ISO-2022-jp','7bit','text/plain');
  $send->Send($message);
  $send->Attach({file => $file});
  $send->Close;
  ##  一時ファイルの削除
  rmtree($file);
}else{
  $send->MailMsg({
    smtp => 'localhost', to => $to, cc => $cc,
    from => $from, charset => 'ISO-2022-jp',
    subject => $subject, msg => $message})
}

## 結果の表示
print header(-charset=>'x-euc-jp'), start_html;
if ($Mail::Sender::Error){
  print
    h3('メールの送信は次のエラーにより失敗しました。'),
    $Mail::Sender::Error,"¥n";
}else{
  print h3('メールを送信しました。');
}

print end_html;
```

また、各パートごとに「Content-type:」を指定して文字コードを指定するために、OpenMultipartをはじめとする一連のメソッドを利用し、最初パートの本文では日本語を送れるように「ISO-2022-JP」（7bitのJISコード）を指定し、次のパートの添付ファイルではMIMEのBase64でエンコードされた「application/octet-stream」を指定している。MailFileを使わなかったのは、このようにパートごとに「Content-type:」を指定したかったからだ。

ファイルを添付するための処理

CGIでアップロードされた添付ファイルは、フォーム変数にファイルハンドルが渡される。そこで、ローカルのファイルを一時的にオープンして、File::Copyモジュールを使ってファイルハンドルを指定して内容をローカルファイルにコピーする。コピーしたファイルを送信してから一時ファイルを削除している。サンプルコードを簡単にするために、ローカルの一時ファイルはmail-sender.cgiと同じ場所に作成される。

エラー処理

最後に、メールが正しく送信された場合はその旨を表示し、正しく送信できなかった場合にはエラーであることを表示するとともに「$Mail::Sender::Error」の内容を表示する。

なお、サンプルコードは短く簡単にするために、メールの送信可／不可以外のエラー処理は特に行っていないことを最後にお断りしておく。実際には、もっとしっかりとしたエラー処理が必要になるだろう。

```
M:¥yasuda¥200304¥CPAN¥DoNotUse>type mail.txt
Return-Path: myname@mydomain
Received:  (省略)
To: yourname@yourdomain
Cc: myfriends@mydomain
From: =?ISO-2022-JP?B?GyRCPnBKczU7OCYbKEI=?= <myname@mydomain>
Subject: =?ISO-2022-JP?B?QOdJGyRCJEclYSE8JWtBdz8uGyhC?=
Date: Tue, 8 Apr 2003 20:51:49 +0900
X-Mailer: Perl script "mail-sender.cgi"
        using Mail::Sender 0.8.00 by Jenda Krynicky
        running on localhost (127.0.0.1)
        under account "root"
Message-ID: <20030408_115149_037444.myname@mydomain>
MIME-Version: 1.0
Content-type: multipart/mixed;
        boundary="Message-Boundary-by-Mail-Sender-1049802709"

This message is in MIME format. Since your mail reader does not understand
this format, some or all of this message may not be legible.

--Message-Boundary-by-Mail-Sender-1049802709
Content-type: text/plain; charset=ISO-2022-jp
Content-description: Mail message body
Content-transfer-encoding: 7bit
Content-disposition: inline

これはテストメールです。
Mail::Senderモジュールを使って、
CGIで添付ファイル付きのメールを送信します。

--Message-Boundary-by-Mail-Sender-1049802709
Content-type: application/octet-stream; name="C:¥tmp¥file.pdf"
Content-transfer-encoding: Base64
Content-disposition: attachment; filename="C:¥tmp¥file.pdf"

JVBERi0xLjIgDSXi48/TDQogDTggMCBvYmoNPDwNL0xlbmd0aCA5IDAgUg0vRmlsdGVyIC9GbGF0
ZURIY29kZSANPj4Nc3RyZWFtDQpIiY3XS24cNxAG4BPMHbh0FpZYfBMOAqjnESCLLOK5gWEFMSwY
8cbXT50sZnPc1VUNAQNb+ucbssguckDB/PPjH3UyOj0IFV1+cgr0k1FGz//68UW9naCHpvuS82b+
```

図3　mail-sender.cgiで送信されるメールの内容例

Mail::Sender.pmモジュール

Eメールの送信

【カテゴリー】メールとニュース
【バージョン】0.8.06
【作者】Jan Krynicky氏
【URL】http://search.cpan.org/author/JENDA/Mail-Sender-0.8.06/

Mail::Sender.pmのココがスゴイ!

Mail::Sender.pmは、メールの送信を行うモジュールである。通常のメールだけでなく、1つまたは複数の添付ファイルを添付したマルチパートのメールを送信することもできる。

実際のSMTP(Simple Mail Transfer Protocol)の仕様やSMTPサーバーの動作を意識することなく、普通のメーラーを使うかのように(場合によってはある意味でメーラーよりも使いやすい!)必要な情報を指定するだけで、簡単にメールを送信できる非常に便利なモジュールだといえる。

実際にMail::Senderが行っている動作を全部Perlで自分で書こうとすると、図1や図3にあるようなメールヘッダーの生成、SMTPサーバープログラムへのデータの引き渡しなど、SMTPに関する正しい知識が必要になるだけでなく、非常に煩雑な作業が必要になり、コードは長くなってしまう。さらに添付ファイルメールの送信となると、ファイルのエンコードから、マルチパートヘッダーの生成などまで処理する必要があり、さらに複雑になる。それらの処理をまったく意識しなくてもPerlから簡単にメールを送信できるというのは非常にありがたい。

Mail::Sender.pmのメソッド(1)

Mail::Senderモジュールには多くのメソッドがあるが、最初のサンプルプログラム(コード1)で示した添付ファイルがある場合に使うMailFileメソッドと、添付ファイルがない場合に使うMailMsgメソッド(実際はこちらを使うことの方が多いだろう)を知っておけばたいていは困らないだろう。

MailMsgメソッドとMailFileメソッドはパラメーターを指定して呼び出すことでメール送信まですべて行ってくれる。ここで紹介するもの以外にもMail::Senderにはさらに高度な機能があるが、それらについてはドキュメントを参照してほしい。

MailMsgメソッド

Eメールを送信する。

【構文】

```
MailMsg([from [, replyto [, to [, smtp [, subject [,
headers]]]]]], message)
```

【引数】

・from
送信元のEメールアドレス。

・replyto
返信先のEメールアドレス。

・to
送信先のEメールアドレス。
「,」で区切って複数指定可能。

・smtp
送信に利用するSMTPメールのFQDN名(ドメイン名を含む完全なホスト名)もしくはIPアドレス。

・subject
件名。

・headers
オプションのヘッダー指定。

・message
メール本文(テキスト)。

【名前付きパラメーター】

上記の各パラメーターは「from =>」のように直接指定してもよい。また下記のパラメーターも名前付きパラメーターとして指定できる。

・fake_from
受信側のメーラーに表示される送信元Eメールアドレス。

・fake_to
受信側のメーラーに表示される送信先Eメールアドレス。

・cc
カーボンコピーを送信するEメールアドレス。
「,」で区切って複数指定可能。

・fake_cc
受信側のメーラーに表示されるカーボンコピー先Eメールアドレス。

・bcc
ブラインドカーボンコピーを送信するEメールアドレス。
「,」で区切って複数指定可能。

・boundary
マルチパートメールの区切り文字列。
指定しない場合は自動的に生成される。

・multipart
　マルチパート部分のMIMEタイプ。
・type
　マルチパートメールのMIMEタイプ。
・ctype
　通常のメールのMIMEタイプ。
　通常は「text/plain」を使う。
・encoding
　メールのコンコード形式。
・charset
　メールの文字コード。
　通常は7ビットJISコードのISO-2022-JPを使う。
・client
　送信元のホスト名。
・priority
　優先度の指定。

MailFileメソッド

　ファイルを添付してEメールを送信する。

【構文】

```
MailFile([from [, replyto [, to [, smtp [, subject [,
headers]]]]]], message, files)
```

【引数】

・files
　添付するファイル名。「,」で区切って複数指定可能。

　files以外の引数はMailMsgメソッドの場合と同じ。
　また、MailMsgメソッドの場合と同じ名前付きパラメーターも使用できる。

Mail::Sender.pmのメソッド（2）

　MailMsgとMailFileは、1つのメソッドでメール全体を送信できるが、マルチパートのメールの各パートのヘッダーを細かく指定したいときなどは、複数のメソッドを組み合わせてメールを送信できる。実用的なサンプル（コード3）のように、細かく文字コードなどを指定する場合には、こちらの方法を使うことになるだろう。
　次に説明するメソッドは、それぞれ単体で使うものではなく、組み合わせて順番に使うものだ。次に示すようなメソッドを組み合わせて使うことで、メール送信の各段階を細かくコントロールできる。

1.OpenMultipartメソッドで送信を開く。
2.Bodyメソッドで本文用ヘッダーを出力する。
3.Sendメソッドで本文を出力する。
4.Attachメソッドで添付ファイルを出力する。
5.Closeメソッドで実際にメールを送信する。

OpenMultipartメソッド

　マルチパートメールの送信を開始する。

【構文】

```
OpenMultipart([from [, replyto [, to [, smtp [, subject
[, headers [, boundary]]]]]]])
```

【引数】

　引数はMailMsgメソッドの場合と同じ。
　また、MailMsgメソッドの場合と同じ名前付きパラメーターも使用できる。

Bodyメソッド

　マルチパートメールの本文のヘッダーを送信する。

【構文】

```
Body([charset [, encoding [, content-type]]])
```

【引数】

　引数はMailMsgメソッドの場合と同じ。
　また、MailMsgメソッドの場合と同じ名前付きパラメーターも使用できる。

Sendメソッド

　メールの内容を送信する。

【構文】

```
Send(@strings)
```

【引数】

・@strings
　送信する内容。

Attachメソッド

　MIMEマルチパートメールの中の1つのパートとしてファイルをエンコードして送信する。

【構文】

```
Attach(description, ctype, encoding, disposition, file)
```

【引数】

・description
　このパートのタイトル。
・disposition
　このパートのdisposition。
　指定しなければ「attachment; filename=ファイル名」となる。
・ctype encoding file
　MailMsg、MailFileメソッドの場合と同じ。
　また、MailMsgメソッドの場合と同じ名前付きパラメーターも使用できる。

Closeメソッド

　メールを終了して送信する。

【構文】

```
Close
```

MIME::Base64.pmモジュール

MIME Base64形式のエンコード・デコード

【カテゴリー】メールとニュース
【バージョン】2.18
【作者】Gisle Aas氏
【URL】http://search.cpan.org/author/GAAS/MIME-Base64-2.18/

MIME::Base64.pmのココがスゴイ!!

データ列をMIME Base64形式の7ビットのテキストコードにデコード／エンコードするためのモジュールだ。マルチパート形式のメールを処理するためには必須のモジュールといえる。Perl 5.8では標準で利用できるようになっている。

MIME::Base64.pmのメソッド

encode_base64メソッド

指定した文字列をBase64でエンコートした結果を返す。

ファイルハンドルを指定すると、ファイルの内容をエンコードして返す。

【構文】

encode_base64(変換元文字列)

decode_base64メソッド

指定した文字列をBase64でデコードした結果を返す。

ファイルハンドルを指定すると、ファイルの内容をデコードして返す。

【構文】

decode_base64(変換元文字列)

Mail::CheckUser.pmモジュール

Eメールアドレスのチェック

【カテゴリー】メールとニュース
【バージョン】1.19
【作者】Ilya Martynov氏
【URL】http://search.cpan.org/author/ILYAM/Mail-CheckUser-1.19/

Mail::CheckUser.pmのココがスゴイ!!

最後に、メール関係のモジュールでちょっとしたものを追加で説明しておく。このMail::CheckUserモジュールは、指定したメールアドレスが正しいかどうかを調べる。メールアドレスの記述形式のチェックといった簡単なことから、実際に送信しようとしているメールサーバーが存在するか、そのサーバーがメールを受け付けているかといったチェックを行うことができる。今回のサンプルコードでは行っていないが、現実には必要となるエラーチェックを行うために使うと便利だ。

Mail::CheckUser.pmのメソッド

check_emailメソッド

メールアドレスが正しいかどうかを確認する。

【構文】

check_email($email)

【引数】

・$email

チェックするメールアドレス。

【戻り値】

trueが返されるとそのメールアドレスは正しい。

last_checkメソッド

check_emailメソッドで確認した結果の詳しい情報を得る。

【構文】

last_check

【戻り値】

{ ok => OK, code => CODE, reason => REASON }

という形のハッシュ参照が返される。

・CODE

確認の結果を示すコードの定数。

・OK

trueならばメールアドレスは正しい。

・REASON

確認の結果を示すテキスト。

CODEで返される定数や、Mail::CheckUserモジュールが利用する環境変数などについてはドキュメントを参照してほしい。

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp