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5回

Text：情報技研

Mail::モジュールを使ったメールの受信処理

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/

Mail::POP3Clientモジュールを使ってメールを受信しよう

前回はMail::Senderモジュールを使ってメールの送信を簡単に行う方法を解説し、CGIモジュールと組み合わせてCGIフォームからメールの送信を行った。では、送信を行ったのだから次は受信というのが普通の考え方だろう。そこで、今回はメールの受信を行うモジュールについて解説していく。

まず、メールの受け取り処理といってもいくつかの方法がある。メールを送る場合は、メールを作成してサーバーに投げつければよいだけなのでわかりやすい。では逆に受け取る場合はどうだろうか。まず送られたメールをどのように受け取るかで処理が異なってくるだろう。送信されたメールの到着をきっかけとしてメールを処理するのか、それともサーバーに到着しているメールを取りに行くことでメールを処理するのかで大きく異なる。前者であれば、メールサーバーとの連携が必要であるが、それを除けばリアルタイムに1通単位での処理ができる。後者はメールサーバーとの連携は必要ないが、サーバーに蓄積されているメールの処理となるので複数のメールの処理が必要となる。また、サーバーにメールを取りに行くのは受信者のアクションであるため、リアルタイムにメールの処理を行うためには常に到着のチェックが必要になる。

とはいえ、到着メールをリアルタイムで処理するためにはメールサーバーとの連携が必要になるので、今回は一般的な環境でも簡単に行える、メールを取りに行く受信方法から解説する。

POPサーバーとの接続

Mail::POP3Clientモジュールは、到着したメールが蓄積されているPOPサーバーからメールを取り出すモジュールである。前回紹介したMail::Senderと同様に、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、APOP認証が必要なPOPサーバーにアクセスする際にはMD5モジュールも必要になる。

メールの受信と言っても、メールをどう処理するかでその内容はさまざまである。そこで、まずPOPサーバーに接続することから始め、次に、到着しているメッセージの数の表示、そして実際のメッセージ内容の表示と順番に行う。それでは、Mail::POP3Clientモジュールを使って、POPサーバーに接続してみよう。

コード1を実行するとPOPサーバーに接続できているかどうかを確認できる。図1で示した実行結果ではメールサーバー「localhost」に対してユーザー「user」とパスワード「password」でメールを確認できている。実際の応答はサーバーの種類によって異なるので、この結果のとおりになるとは限らない。

まずは、このサンプルコードを実行して問題なくPOPサーバーに接続できるかどうかを確認する。このサンプルコードはデバッグモードでPOPサーバーへの接続と切断を行っているだけである。デバッグモードの指定は名前付きパラメーター「DEBUG=>1」で行っている。POPサーバー名、ユーザー名、パスワードを正しく指定してもうまく接続できない場合、「AUTH_MODE」を「BEST」

コード1　POPサーバー接続のサンプルコード（mail-pop3client-1.pl）

```
#!/usr/bin/perl
use Mail::POP3Client;

$pop3 = new Mail::POP3Client(
  USER=>'user', PASSWORD=>'password',
  HOST=>'localhost', AUTH_MODE=>'BEST',
  DEBUG=>1);

$pop3->Close();
```

```
POP3 <- +OK Qpopper (version 4.0.4) at localhost starting.  <95256.
1053035632@localhost>
 at ./mail-pop3client-1.pl line 4
POP3 -> USER user
 at ./mail-pop3client-1.pl line 4
POP3 <- +OK Password required for user.
 at ./mail-pop3client-1.pl line 4
POP3 -> PASS password
 at ./mail-pop3client-1.pl line 4
POP3 <- +OK user has 2 visible messages (0 hidden) in 2239 octets.
 at ./mail-pop3client-1.pl line 4
POP3 -> STAT
 at ./mail-pop3client-1.pl line 4
POP3 <- +OK 0 0
 at ./mail-pop3client-1.pl line 4
POP3 -> QUIT
 at ./mail-pop3client-1.pl line 8
POP3 <- +OK Pop server at localhost signing off.
 at ./mail-pop3client-1.pl line 8
```

図1　mail-pop3client-1.plの実行結果例

ではなく「APOP」か「PATH」に変えて試してみよう(AUTH_MODEパラメーターについては後述のメソッドの解説を参照)。

POPサーバーにあるメッセージの確認

次に、デバッグモードではなく通常のモードで接続して、まずは簡単にPOPサーバー上にあるメッセージ数を確認する。コード2を実行すると、現在到着しているメッセージ数とそれぞれのメッセージのサイズが表示される(図2)。

コード2　到着メール数を確認するサンプルコード(mail-pop3client-2.pl)

```
#!/usr/bin/perl
use Mail::POP3Client;

$pop3 = new Mail::POP3Client(USER=>'user',
  PASSWORD=>'password', HOST=>'localhost',
  AUTH_MODE=>'BEST');

print
  'You have ', $pop3->Count(), " message(s).¥n",
  "# size(bytes)¥n".$pop3->List();

$pop3->Close();
```

```
You have 2 message(s).
# size(bytes).
1 553
2 1686
```

図2　mail-pop3client-2.plの実行結果例

コード3　到着メッセージの内容を表示するサンプルコード(mail-pop3client-3.pl)

```
#!/usr/bin/perl
use Mail::POP3Client;

$pop3 = new Mail::POP3Client(USER=>'user',
  PASSWORD=>'password', HOST=>'localhost',
  AUTH_MODE=>'BEST');

for ($i = 1; $i <= $pop3->Count(); $i++){
    print "¥n--[No.$i]--------------------------------¥n";
    foreach ($pop3->HeadAndBody($i)){print $_,"¥n"};
}

$pop3->Close();
```

```
--[No.1]--------------------------------
Return-Path: <myname@mydomain>
Received:  (省略)
Message-Id: <200305152209.h4FM96cN095367@mydomain>
To: user@mydomain
Subject: Test Mail
Content-Type: text/plain; charset=US-ASCII
Date: Fri, 16 May 2003 01:15:37 +0900
From: myname@mydomain

This is Test Mail

--[No.2]--------------------------------
Return-Path: <c>
Received:  (省略)
Message-Id: <200305161715.CAA60712@example>
To: user@mydomain
Subject: Welcome to Perl Mail!
Content-Type: text/plain; charset=US-ASCII
Date: Fri, 16 May 2003 00:48:25 +0900
From: 200305161715.CAA60712

How is Perl programming?
```

図3　mail-pop3client-3.plの実行結果例

POPサーバー上のメッセージの表示

次は、実際に到着しているメッセージを表示する。コード3を実行すると、図3のように現在到着しているメッセージの番号と内容をそのまま順に表示していく。

ここでの処理としては、サーバー上にあるメッセージの数を調べ、メッセージ番号を指定してヘッダーと本文を取得してそれを表示するものである。

さまざまなモジュールを使ってメールの処理を行おう

Mail::POP3Clientモジュールは、POPサーバーに蓄積されたメールのヘッダー部分と本文の部分をそのまま読み出すだけで、それ以上の処理を行うことはできない。実際に受信メールの処理を行う際には、メールアドレスやサブジェクトなどのヘッダー部分の処理やマルチパートの処理を行う必要があるだろう。

メールヘッダーの処理

まずはメールのヘッダー処理を行うためにMail::Toolsの中のMail::Headerモジュールを使ってみる。Mail::Toolsモジュールとはメールに関する処理を行うモジュール群である。

サンプルコード4では、Mail::POP3Clientモジュールで取り出したメールのヘッダーをMail::Headerモジュールに渡している(その部分をサンプルコード上に示してある)。渡したヘッダー部分から、Mail::Headerモジュールのgetメソッドを使ってヘッダー部分を取り出してメッセージごとに表示している(図4)。

コード4　ヘッダーの取り出しを行うサンプルコード（mail-header.pl）

```
#!/usr/bin/perl
use Mail::POP3Client;
use Mail::Header;

$pop3 = new Mail::POP3Client(USER=>'user',
  PASSWORD=>'password', HOST=>'localhost',
  AUTH_MODE=>'BEST');
$header = new Mail::Header;

for ($i = 1; $i <= $pop3->Count(); $i++){
  $header->extract([$pop3->Head($i)]); #ヘッダーの受け渡し
  print "[Message No.$i]¥n",
    "日付:¥t¥t", $header->get(Date),"¥n",
    "題名:¥t¥t", $header->get(Subject),"¥n",
    "発信者:¥t", $header->get(From),"¥n";
}

$pop3->Close();
```

```
[Message No.1]
 日付:          Sun, 4 May 2003 19:53:06 +0900
 題名:          Test
 発信者:        myname@mydomain
[Message No.2]
 日付:          Wed, 7 May 2003 07:22:49 +0900
 題名:          This is Test Mail
 発信者:        Inst of Info Tech <yourname@yourdomain>
[Message No.3]
 日付:          Mon, 12 May 2003 18:16:41 +0900
 題名:          =?ISO-2022-JP?B?GyRCRnxLXDhsJT8lJCVIJWskRyRiGyhCTOs=?=
 発信者:        Inst of Info Tech <contact@iitdomain>
[Message No.4]
 日付:          Tue, 13 May 2003 00:00:29 +0900
 題名:          =?ISO-2022-JP?B?GyRCJUYlOSVIJWEhPCVrGyhC?=
 発信者:        =?ISO-2022-JP?B?GyRCPnBKczU7OCYbKEI=?= <contact@iitdomain>
[Message No.5]
 日付:          Thu, 15 May 2003 12:01:33 +0900
 題名:          =?ISO-2022-JP?B?GyRCJUYlOSVIJWEhPCVrGyhC?=
 発信者:        info@iitdomain (=?ISO-2022-JP?B?GyRCPnBKczU7OCYbKEI=?=)
```

図4　mail-header.plの実行結果例

コード5　ヘッダーを取り出して見やすく表示するサンプルコード（mail-header.pl）

```
#!/usr/bin/perl
use Mail::POP3Client;
use Mail::Header;
use Mail::Address;
use Jcode;
use Date::Parse;

$header = new Mail::Header;

$pop3 = new Mail::POP3Client(USER=>'user',
  PASSWORD=>'password', HOST=>'localhost',
  AUTH_MODE=>'BEST');

for ($i = 1; $i <= $pop3->Count(); $i++){
  $header->extract([$pop3->Head($i)]);
  ($sec,$min,$hour,$day,$month,$year,$zone)
          = strptime($header->get(Date));
  $date = sprintf ("%04d/%02d/%02d %02d:%02d:%02d¥n",
          $year +1900, $month, $day, $hour, $min, $sec);
  @addr = Mail::Address->parse(
          jcode($header->get(From))->mime_decode()->euc);

  print  "[Message No.$i]¥n",
    "日付:¥t¥t", $date,"¥n",
    "題名:¥t¥t",
     jcode($header->get(Subject))->mime_decode()->euc,"¥n",
    "発信者:¥t", $addr[0]->address(),"¥n",
    "発信者名:¥t", $addr[0]->name(), "¥n",
}

$pop3->Close();
```

ヘッダー情報の日本語処理

図4の実行結果を見ると、メールの題名やメールアドレスに含まれている日本語がエンコードされたままの状態で表示されている。これでは何が書かれているのかわからないので、日本語を正しく表示するための処理が必要になる。これは過去に何度も説明しているので、Jcodeモジュールを使えばいいことはすぐに思いつくだろう。また、RFC822準拠の日付表示もそのままではあまり見栄えはよくない。発信者の情報もメーラーなどによってその書式はまちまちである。そこで次にいくつかのモジュールを使って、これらの表示を見やすくする（コード5）。

実行結果（図5）を見るとわかるように日本語が正しく表示され、日付もシンプルな表記に変換され、発信者も名前とアドレスが分離されて見やすく表示されている。これを実現するために、日本語処理ではおなじみのJcodeモジュールのほかに、アドレスの解析をするMail::Addressモジュール、日付の文字列を解析するDate::Parseモジュールを利用している。

```
[Message No.1]
 日付:          2003/04/04 19:53:06
 題名:          Test
 発信者:        myname@mydomain
 発信者名:
[Message No.2]
 日付:          2003/04/07 07:22:49
 題名:          This is Test Mail
 発信者:        yourname@yourdomain
 発信者名:      Inst of Info Tech
[Message No.3]
 日付:          2003/04/12 18:16:41
 題名:          日本語タイトルでもOK
 発信者:        contact@iitdomain
 発信者名:      Inst of Info Tech
[Message No.4]
 日付:          2003/04/13 00:00:29
 題名:          テストメール
 発信者:        contact@iitdomain
 発信者名:      情報技研
[Message No.5]
 日付:          2003/04/15 12:01:33
 題名:          テストメール
 発信者:        info@iitdomain
 発信者名:      情報技研
```

図5　mail-header2.plの実行結果例

サンプルコードでは、Date::Parseモジュールのメソッドstrptimeで日付の文字列からその内容を解析して変数に分割代入している。

アドレスの処理に使ったMail::Addressモジュールは、parseコンストラクターにメールアドレスの文字列を与えることで、1つ1つのアドレスのオブジェクトを配列として返す。この例ではFrom行の解析を行っているので、配列の最初のオブジェクトに対してaddressメソッドでメールアドレスを取り出し、nameメソッドで名前を取り出して表示している。

Mail::POP3Client.pmモジュール

【カテゴリー】メールとニュース
【バージョン】2.14
【作者】Sean Dowd氏
【URL】http://search.cpan.org/author/SDOWD/Mail-POP3Client-2.14/POP3Client.pm

POP3サーバーとやり取りを行う

Mail::POP3Client.pmのココがスゴい!

RFC 1939に従ってPOP3サーバーとやり取りを行うことで、サーバーに蓄積されている電子メールのメッセージを簡単に読み出せるモジュールだ。

Mail::POP3Client.pmのメソッド

newコンストラクター

POP3接続を新たに生成して指定されたサーバーに接続する。

【構文】

```
new Mail::POP3Client(USER, PASSWORD[, HOST, PORT, DEBUG, AUTH_MODE]);
```

【引数】

・USER
ユーザー名。
・PASSWORD
パスワード。
・HOST
POP3サーバーのホスト。
・PORT
POPポート番号の指定。
・DEBUG
正の正数をセットするとデバッグモードになる。
・AUTH_MODE
認証のモードを指定する。指定できるのは「BEST」「APOP」「PASS」。BESTを指定すると、APOP認証を先に行い失敗したらその後平文パスワード認証を行う。明示的に認証方法を指定する場合にはAPOPやPASSを指定する。

【名前付きパラメーター】

上記の各パラメーターは、「USER=>'user'」のようにして名前付きパラメーターとして指定できる。このほかに、サーバーへの接続タイムアウトを「TIMEOUT」で指定できる。

Body(msgnum)メソッド

msgnumで指定されたメッセージの本文を取得する。

BodyToFile(HANDLE, msgnum)メソッド

msgnumで指定されたメッセージの本文を、HANDLEで指定されたファイルに保存する。

Closeメソッド

正しい手順で接続を終了する。

Countメソッド

サーバー上のメッセージ数を返す。

Delete(msgnum)メソッド

msgnumで指定されたメッセージ番号に削除マークを付ける。
実際の削除はCloseメソッドを実行したときに行われる。

Head(msgnum)メソッド

msgnumで指定されたメッセージのヘッダーを取得する。

HeadAndBody(msgnum)メソッド

msgnumで指定されたメッセージのヘッダーと本文の両方を取得する。

Lastメソッド

最後のメッセージ番号を取得する。

Listメソッド

各メッセージのサイズをリストの形で取得する。引数にメッセージ番号を指定して実行すると、<msgnum> <size>という形でメッセージのサイズを返す。

Messageメソッド

最新のステータスメッセージまたはエラーメッセージを取得する。

Resetメソッド

Deleteメソッドで付けた削除マークをすべて消去する。

Size(num)メソッド

numで指定されたメッセージのサイズを返す。

Stateメソッド

現在の接続に関する情報を返す。

Mail::Header.pmモジュール

メールヘッダーの処理

【カテゴリー】メールとニュース
【バージョン】1.58
【作者】Mark Overmeer
【URL】http://search.cpan.org/author/MARKOV/MailTools-1.58/Mail/Header.pm

Mail::Header.pmのココがスゴい!

RFC822準拠のメールヘッダーを読み込んだり、作成したり、書き出したり、操作したりできる。サンプルプログラムでは読み込みしか行っていないが、実際には読み込んだヘッダーを編集して再利用することもできるモジュールだ。Mail::Headerモジュールは Mail::Toolsモジュール群に含まれるモジュールである。

Mail::Header.pmのメソッド

newコンストラクター

ファイル記述子か配列への参照を指定した場合には、新しいオブジェクトのヘッダーを指定された配列かファイル記述子から読み込まれた情報で初期化する。指定がない場合にはオブジェクトのみを生成する。

【構文】

```
new Mail::Header([ARG], [OPTIONS]);
```

【引数】

ARGにファイル記述子か配列への参照を指定する。

【名前付きパラメーター】

オプションとして「Modify」「MailFrom」「FoldLength」の各名前付きパラメーターを指定できる(詳細はドキュメントを参照)。

その他の主なメソッド

- `fold([LENGTH]);`
  LENGTHで指定した長さでヘッダーを折り返す。
- `extract(ARRAY_REF);`
  指定した配列からヘッダーを抜き出してオブジェクトを返す。
- `read(FD);`
  指定したファイル記述子からヘッダーを読み込む。
- `add();`、`replace();`
  ヘッダーの追加、置換を行う。
- `combine(TAG);`
  指定したヘッダータグのインスタンスをまとめる。
- `get(TAG);`
  指定したヘッダータグを取得する。
- `delete(TAG);`
  指定したヘッダータグを削除する。
- `unfold(TAG);`
  指定したヘッダータグの折り返しを解除して1行にする。

Mail::Address.pmモジュール

メールアドレスの処理

【カテゴリー】メールとニュース
【バージョン】1.58
【作者】Mark Overmeer
【URL】http://search.cpan.org/author/MARKOV/MailTools-1.58/Mail/Address.pm

Mail::Address.pmのココがスゴイ!!

RFC822準拠のメールヘッダーの中のメールアドレス部分を処理できる。サンプルプログラムでは読み込みしか行っていないが、生成にも使えるモジュールである。Mail::Addressモジュールも Mail::Toolsモジュール群に含まれるモジュールだ。

Mail::Address.pmのメソッド

newコンストラクター

指定した要素をもつメールアドレスのオブジェクトを新しく生成する。各パラメーターには「ADDRESS=>'addr'」のようにして名前付きパラメーターを指定できる。

【構文】

```
Mail::Address->new(PHRASE,ADDRESS[,COMMENT]);
```

【引数】

実際のアドレスにおいて次のどちらかの形式となる。

・PHRASE <ADDRESS> (COMMENT)

・ADDRESS (COMMENT)

parseコンストラクター

指定した行を解析したオブジェクトのリストを返す。指定する行は「To」「Cc」「Bcc」「From」などのヘッダー部分である。1つのアドレスに対して1つのオブジェクトが対応し、アドレスは1つとは限らないため、リスト形式でオブジェクトが返される。

【構文】

```
Mail::Address->parse($line);
```

その他の主なメソッド

・phrase()

フレーズ部分を返す。

・address()

アドレス部分を返す。

・comment()

コメント部分を返す。

・name()

オブジェクト内の情報から適切な識別名を返す。

・host()

アドレスの@より後ろの部分を返す。

・user()

アドレスの@より前の部分を返す。

Mail::Toolsモジュール群のその他のモジュール

今回は、Mail::Header、Mail::Addressを紹介したが、Mail::Toolsモジュール群にはこれ以外にもMail::Internet(メールメッセージの処理)、Mail::Mailer(メールサーバーへのインターフェイス)、Mail::Send(メールサーバーへのインターフェイス)など多くのモジュールがまとめられている。

URL http://search.cpan.org/author/MARKOV/MailTools-1.58/

Date::Parse.pmモジュール

日付の文字列を解析して時間値を分割、変換

【カテゴリー】データとデータ型
【バージョン】1.14
【作者】Graham Barr
【URL】http://search.cpan.org/author/GBARR/TimeDate-1.14/lib/Date/Parse.pm

Date::Parse.pmのココがスゴイ!!

str2timeとstrptimeの2つのメソッドが提供されている。多国語もサポートしており、言語を指定すれば文字列をその言語で解析できる。

Date::ParseモジュールはTimeDateモジュール群に含まれるモジュールである。

Date::Parse.pmのメソッド

str2timeメソッド

DATE文字列を解析してUNIXの時間値を返す。

【構文】

```
str2time(DATE [,ZONE]);
```

strptimeメソッド

DATE文字列を解析して(秒,分,時,日,月,年,ZONE)という数値配列に代入する。

【構文】

```
strptime(DATE [,ZONE]);
```

Date::Formatモジュール群のその他のモジュール

Date::Formatモジュール群には、Date::Parse.pmモジュールのほかにも、以前紹介したDate::Formatモジュールなど、さまざまな日付処理のモジュールが含まれる。

URL http://search.cpan.org/author/GBARR/TimeDate-1.14/1236

［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