use Perl::Object qw(1);

文 編集部 mitty

TIMTOWTDI[*1]

お久しぶりです。ってことで、Perl です。前回から一年以上[*2] 経ってしまいましたが、めげずに再開したいと思います。なお、本記事は **MOTTAINAI** の精神から、過去の記事[*3] を~~再利用~~適宜引用して構成されています。

過去の記事は WORD Press から、どうぞご覧下さい。

http://www.word-ac.net/

Perl の処理系について

Apple の OS X や Linux、FreeBSD など、Windows 以外の一般的な OS であればおそらく最初から Perl が導入されているか、あるいは OS それぞれに標準の導入方法があるはずです。

Windows については ActivePerl や Strawberry Perl、そして Cygwin[*4] 上の perl などが利用出来ます。もちろんソースコードから自分でビルドすることも出来るでしょう。

なお、筆者は主に perl 5.10 以降でコードを書いたりモジュールを使用したりしています。本記事で扱っている内容はおそらく perl 5.8 でも動くことと思いますが、確認をしていませんのであしからずご了承下さい。システムに導入されている perl のバージョンが古い(5.8.x)等の問題があれば perlbrew[*5] や cpanm[*6] を利用することを検討するのも良いでしょう。

Perl や perlbrew などの導入については記事の内容から離れてしまうので、ここでは触れません。詳しくは Google 先生などを活用して下さい。

use による既存モジュールの活用

さて、それでは本題である Perl でのオブジェクト指向について触れていくことにしましょう。

---

*1 TIMTOWTDI："There's more than one way to do it." のことで Perl のモットーだそうです。最近**拡張され**、"There's more than one way to do it, but sometimes consistency is not a bad thing either（**TIMTOWTDIBSCINABTE**）."になったらしい。"*Tim Toady Bicarbonate*"って発音してね、って**ウィキペたんが言ってた。**

*2 前回から一年以上：2011 年 2 月発行の 17 号に掲載の「use Perl::Object 0;」から数えて。

*3 過去の記事：上記 17 号と、その一つ前の 16 号を参照下さい。

*4 Cygwin：Windows 上で動作する UNIX ライクな環境です。ls や vim といった、Linux などでおなじみのコマンドが利用出来るようになります。

*5 perlbrew：管理者権限を必要とせず、ユーザのホームディレクトリに任意のバージョンの perl を導入し切替えながら利用出来るコマンドのこと。

*6 cpanm：CPAN(後述)からソースコードを取得し、ビルドしてインストールしてくれる賢いスクリプトです。cpan minus の略と言うことになっていますが、全然マイナスじゃない気がします。

まずは CPAN[*7] などから既存のモジュールを導入して利用してみようと思います。

CPAN モジュールの導入

少し前の号の記事[*8] でも触れられていましたが、CPAN には当然のように Twitter 関連のモジュールがあります。http://search.cpan.org/で「Twitter」と検索してみると、140 件ほどヒットすると思います。

CPAN の既存モジュールが非常に多いことからも気づかれていることと思いますが、Perl では「やりたいこと」に対して、大抵の場合使えそうなモジュールの候補が沢山ヒットします。そして、モジュールの説明ページの「DESCRIPTION」を見ても、どれを選んで良いか分からない場合が結構あります。そのような時は、ページ右上にある「Dependencies」を開いてみましょう。そのモジュールが依存しているモジュールが、ツリー構造で表示されます。

様々なモジュールを次々と読み込み組み合わせて使用するというのが、ある意味 Perl モジュールの特徴となっているので、この Dependencies を参考に安全そうなもの[*9] を選ぶと良いでしょう。

さて、Twitter 関連のモジュールですが、よく使われるものに「Net::Twitter」や「Net::Twitter::Lite」、あるいは「AnyEvent::Twitter」などがあります。UserStream を使ったりエラー処理を細かく行う必要がない場合は、依存するモジュールが少ない Net::Twitter::Lite を用いるのが良いのではないでしょうか。

cpanm が使用出来る環境であれば cpanm を、そうでなければ cpan を用いて、次のように入力することで Net::Twitter::Lite がインストールされます。なお、シェルのプロンプトから分かるように、perlbrew を利用してホームディレクトリに導入することを想定しています。システム全体に導入する場合は、cpan コマンドを使用するよりも Linux のディストリビューションごとに用意されているパッケージ管理ソフトからインストール[*10] する方が良いかも知れません。

```
$ cpanm install Net::Twitter::Lite
      または
$ cpan install Net::Twitter::Lite
```

cpanm であれば、依存するモジュールは自動的に導入されるはずです。cpan の場合は、依存するモジュールを導入するかどうかたずねてくる場合があるようです。その場合は、y と答えてイン

---

*7 CPAN：「Comprehensive Perl Archive Network」の略語で、Ruby における RubyGems や PHP における PEAR に近いでしょうか。2012/05/14 現在、「106536 Modules, 9684 Uploaders」だそうです。まさに玉石混淆でカオス。一年前(2011/01/21)から実に 17180 モジュールも増えているのか……。

*8 少し前の号の記事：19 号の「全自動ふぁぼ bot プロジェクト」で、Perl を用いて書かれた bot が紹介されています。

*9 安全そうなもの：新しいモジュールやきちんとメンテナンスされているモジュール、検索結果で上位に来るモジュールなどであれば、ビルドに失敗したり API の仕様が古めかしかったり、思わぬバグが潜んでいたりする可能性が低いです。モジュールごとにいろんな環境でのビルド成功率も表示されるので、それによって判断するのも有りでしょう。

*10 パッケージ管理ソフトからインストール：例えば Debian 6.0 だと「libnet-twitter-lite-perl」という名前で用意されています。ただし、バージョンが若干古い場合が(まれによく)あります。

ストールしましょう。いずれのコマンドも、モジュールの依存関係自体は自動で解決してくれます。

ActivePerl については、選択出来るモジュール数は CPAN 全体よりもずっと減ってしまいますが、ppm という GUI ツールがあります。ppm からインストール出来るモジュールは ppm 経由でインストールする方が確実です。

モジュールの使い方

モジュールの使い方も Net::Twitter::Lite を例に説明しようと思ったのですが、SYNOPSIS[*11] のコードがちょっと古くて参考にならないことと、OAuth[*12] 周りの話が面倒なので、代わりに Net::DNS を例に説明したいと思います。モジュールの簡単な使い方は、モジュールの説明ページか、以下のコマンドにより表示されるモジュールのドキュメントの SYNOPSIS で説明されています。

```
$ perldoc Net::DNS
```

```
SYNOPSIS
    "use Net::DNS;"
```

さて、SYNOPSIS から分かるのは……。いくらなんでもこれじゃ何も分かりませんね。仕方がないので、以前 open-coins[*13] で必要に迫られて書いたスクリプトを載せてみます。このスクリプトは、ドメインあるいは IP アドレスが一行に一つずつ書かれたテキストファイルを引数に取り、一つ一つ名前解決して結果を表示するスクリプトです。第二引数に DNS サーバが指定されていればそれを対象に名前解決を行います。第二引数は省略可能です。

dnsbench.pl *http://lab.mitty.jp/trac/lab/browser/lab/trunk/misc/dnsbench.pl*

```
1:      #! /usr/bin/perl -w
2:
3:      use strict;
4:      use warnings;
5:
6:      use Net::DNS;
7:
8:      my $hostlist = shift @ARGV or die "usage: $0 list_of_hosts [nameserver]";
```

*11 SYNOPSIS：「概要」のこと。CPAN モジュールの場合は「少し手を加えるだけで使えるコード例」であることが多い。

*12 OAuth：あるサービスに登録されているユーザのアカウントを別のサービスが利用する際に、ログインパスワード等を知らせることなく必要な権限だけ委譲するための仕組み。詳しくは Google 先生にお願いします。

*13 open-coins：情報科学類生有志で運営している、学類公認(公式ではない)の計算資源とスタッフのこと。 http://www.open.coins.tsukuba.ac.jp/

```
9:      if (! -r $hostlist) {
10:         die "$0: cannot read $hostlist";
11:     }
12:     my $nameserver = shift @ARGV;
13:
14:     my $res;
15:     if ($nameserver) {
16:         warn "$0: performing DNS query with server($nameserver)";
17:         $res = Net::DNS::Resolver->new(
18:             nameservers => [$nameserver],
19:         );
20:     }
21:     else {
22:         warn "$0: use system default nameservers";
23:         $res = Net::DNS::Resolver->new;
24:     }
25:
26:
27:     open my $list, '<', $hostlist or die "$0: filename: $!";
28:     while (my $host = <$list>) {
29:         chomp $host;
30:         my $query = $res->send($host);
31:         if ($query) {
32:             print "$host -> ";
33:             if ($res->errorstring ne 'NOERROR') {
34:                 print $res->errorstring, "¥n";
35:                 next;
36:             }
37:             foreach my $rr ($query->answer) {
38:                 if ($rr->type eq 'A') {
39:                     print $rr->address, " ";
40:                 }
41:                 elsif ($rr->type eq 'PTR') {
42:                     print $rr->ptrdname, " ";
43:                 }
44:             }
45:             print "¥n";
46:         }
47:         else {
48:             warn "$0: query failed: ", $res->errorstring, "¥n";
49:         }
50:     }
```

さて、解説をしていきましょう。

インストールされたモジュールは **use** 宣言によって実際に利用出来るようになります。**use** には他にも様々な使い道があるのですが、とりあえずはモジュールをロードするための一文だと思って下さい。

Net::DNS モジュールが提供する機能のうち、リゾルバは Net::DNS::Resolver モジュールから利用出来ます。従ってコンストラクタには **Net::DNS::Resolver->new()** を用いています。
Net::DNS::Resolver モジュールについてはまだ Net::DNS モジュールよりは参考になる SYNOPSIS が存在するので参照してみて下さい。

15 行目で、スクリプト起動時に DNS リゾルバが指定されているかどうかを判断し、指定されている場合はそれを **nameservers** というハッシュキーを用いてコンストラクタの **new** メソッドに指定しています。17-19 行目では **new()** と括弧が付いているのに 23 行目の **new** で括弧が無くなっているのは、Perl では関数やメソッド呼び出しの際、**()** を省略出来るためです。new メソッドにより、**$res** に Net::DNS::Resolver オブジェクトが代入されます。

30 行目にて、一行ごとに **send** メソッドを用いて DNS クエリを送信し、結果を**$query** に代入しています。Net::DNS::Resolver モジュールのドキュメントを読むと、似た機能を持つメソッドに search および query があります。ホスト名のサフィックスに DNS Suffix Search List を用いるかどうかの違いはありますが、いずれも最終的には名前解決をするために send を呼んでいるようです。

send の返値は、DNS リゾルバに接続出来ないなどの場合は **undef**[*14]、それ以外では Net::DNS::Packet オブジェクトとなっています。つまり 31 行目では、エラーではないことを確認しているのです。

DNS クエリ自体の結果ステータスは、元の**$res** に格納されている Net::DNS::Resolver オブジェクトの、**errorstring** メソッドによって得ることが出来ます。**NOERROR** で無ければ何らかのクエリエラーが発生したとしてステータスを **print** し、**next** によって残りの処理を省略して **while** ループの先頭に戻り次のクエリを実行します。

さて、DNS では、一つのドメインに複数のレコードを設定出来るので、クエリの結果もリストとして得られます。**$query** には上記の通り Net::DNS::Packet オブジェクトが格納されていますが、**answer** メソッドにより Net::DNS::RR オブジェクトのリストが得られます。これを 37 行目でリストの要素を一つずつ**$rr** に代入し、**type** メソッドで正引きか逆引きかを判定し、それぞれ対応したメソッドで結果を **print** しています。

ざっと説明してみましたが、いかがでしょうか。既に Perl 以外の言語でオブジェクト指向に触れている方であれば、そんなに難しくないのではないかと思います。

さて、use 宣言があるのは Net::DNS モジュールだけなのに、Net::DNS::Resolver とか Net::DNS::RR とかはどこからやってきたのか、という疑問があるかと思います。説明が前後してしまいますが、実は「Net::DNS」という名前で提供されているモジュールは Net::DNS モジュール単体のみを含むのではなく、それ以外に様々なモジュールが含まれています。どのようなモジュールが含まれているかは、例えば「Net::DNS」の場合では以下の URL から確認することが出来ます。

http://search.cpan.org/dist/Net-DNS/

---

*14 undef：Perl において未定義値であることを示します。NULL や nil に置き換えて考えて貰えると分かりやすいかも知れません。

このページを見れば、上に出てきたもの以外にも様々なモジュールが含まれていることが分かるでしょう。

実行例

コードだけで実行例が無いと寂しいので、載せてみます。

```
$ cat hosts.txt
mitty.jp
lab.mitty.jp
www.google.co.jp
www.google.com
www.tsukuba.ac.jp
www.coins.tsukuba.ac.jp
130.158.86.11
130.158.86.17
130.158.86.19
130.158.86.200
$ ./dnsbench.pl hosts.txt
./dnsbench.pl: use system default nameservers at ./dnsbench.pl line 22.
mitty.jp -> 59.106.180.123
lab.mitty.jp -> 59.106.180.123
www.google.co.jp -> 173.194.38.119 173.194.38.120 173.194.38.127
www.google.com -> 173.194.38.114 173.194.38.115 173.194.38.116 173.194.38.112 173.194.38.113
www.tsukuba.ac.jp -> 130.158.69.246
www.coins.tsukuba.ac.jp -> 130.158.86.1
130.158.86.11 -> lilac-nwh.coins.tsukuba.ac.jp
130.158.86.17 -> NXDOMAIN
130.158.86.19 -> lilac-print2.coins.tsukuba.ac.jp
130.158.86.200 -> NXDOMAIN
```

うまくいっていることが分かると思います。

今後の予定

本当は今回の記事で use 宣言の詳しい解説とモジュールの名前空間についての説明まで行いたかったのですが、睡眠不足で正気度[*15] が下がってきたそろそろ疲れてきたのでこのくらいにしたい

---

*15 正気度が下がって：口にするのもはばかられる某ライトノベルでは「SAN 値が下がる」という表現の方が多い(把握出来る限りでは「正気度」はシリーズ全体でも 15 回くらいなのに対して、「SAN 値」は 9 巻では第一章タイトルに含まれていることもあって奇数ページの左上に毎回出てくるため、それだけで 35 回、それ以外にも 60 回くらい登場)ですが、元となった「クトゥルフの呼び声(TRPG)」では、厳密には下がったり減ったりと変化するのは「現在の正気度」だそうです。まぁ、「SAN 値」の方が語呂(?)が良いですからね、「SAN 値直葬」とか。

と思います。それでは

参考文献

- Larry Wall, Tom Christiansen, Jon Orwant『プログラミング Perl』近藤 嘉雪訳（オライリー・ジャパン 第 3 版 2002 年）
- Randal L. Schwartz, Tom Phoenix（2003）*Learning Perl Objects, References & Modules* O'Reilly Media
- http://www.perl.org/
- Google 先生

また、記事作成の際に参考にしたサイトは以下のページにまとめてあります。

http://lab.mitty.jp/trac/lab/wiki/Commentary/WORD

免責

本記事の内容は、引用部分や素材など、著作権その他の権利が筆者に帰属しない物、あるいは個別に但書きされている物を除き、Perl そのものと同じ「Artistic License[*16]」か、あるいは「2-clause BSD license[*17]」に従って再利用できます。

また、記事タイトルの「Perl::Object」というモジュールは実際に存在しているわけではありませんのでご注意下さい。

蛇足

ちなみに、今更野暮[*18]ですが記事の見出しは「名状しがたい Perl（バール）のようなもの（コード）」のつもりです。バールと Perl の発音が似てるので使ってみただけだったり。

しかし、「這いよれ!ニャル子さん」で「正気度」や「SAN 値」が何回くらい出てきているのか調べたり、「名状しがたい」が H.P.Lovecraft 御大の原文（英語）でどのような単語だったのか[*19]調べるために深夜に冒涜的な英文[*20]と格闘したりすると、急激に正気度が下がりますね。皆さんも気をつけ

本記事の内容は、以下の URL でも公開しております。

http://lab.mitty.jp/word/

*内容に間違い・質問などありましたら、*twitter:@mittyorz *あるいは読者アンケートにて連絡して頂ければ幸いです。*

---

*16 Artistic License：http://dev.perl.org/licenses/artistic.html

*17 2-clause BSD license：http://www.freebsd.org/copyright/freebsd-license.html

*18 今更野暮：ネタや冗談の説明をすることほど興ざめなことはありませんよね。**特に通じなくて解説が必要なときほど。**

*19 どのような単語だったのか：indiscribable の他に、nameless や unnamable も使われていました。

*20 冒涜的な英文：恐るべきことに、その多くはパブリックドメインとしてウェブ上にも公開されています……。