use Perl::Object;

文 編集部 mitty

TIMTOWTDI[*1]

やり方は一つじゃないよね。

ってことで Perl です。**Perl です。**大事なことなので（ｒｙ

なんで今更 Perl なのかと問われても一言では語れないわけですが、まぁ強いて言えばそこに Perl があるから[*2]、でしょうか。

対象~~年齢~~

```
1.      #! /usr/bin/perl -w
2.
3.      use strict;
4.      use warnings;
5.
6.      print for <>;                                          #  mycat.pl
```

これを見て何をしているのか[*3] 理解できることを前提に……するのは読む人を選びすぎている気がするので、

- プログラミングについての基礎知識を持ち、
- Perl に触ったことは無くても $ とか ; とか **@_** とか**変な記号**が多いことを知っていて、
- オブジェクト指向の概念はあるけれども実際にコード書くときに使い込んだことは無いよ

ぐらいの方でもなるべく読めるように書いていくつもりです。というか私自身 Perl でオブジェク

---

*1 "There's more than one way to do it." のことで Perl のモットーとして知られる。

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。略が長すぎて困る人は"*Tim Toady Bicarbonate*"って発音してね、って**ウィキペたんが言ってた。**

*2「可愛いは正義」……じゃなかった、「好きだからは真理」はいい言葉だと思います。**言ってるの主に私だけだけど。**

例：「男の娘になる理由は何があるんでしょうか？女の子じゃダメなんでしょうか？」「だって、男の娘とか好きだから！」「わぁい！」

*3 このコードを「mycat」とか名前を付けて、chmod +x すると*nix での実際の cat っぽくなります。といっても、本家とは違いファイル名以外の引数を取ることはできませんが。

ト指向プログラミングってあんまりやったことが無いので、自分自身のレベルアップ[*4] 的な意味も込めて。

なので、プログラミングは大学の授業(情報科学類の「プログラミング入門 I」とか)が初めて、という方はちょっと辛いかもです。

準備するもの

今回使用する Perl は Perl 5 です。オブジェクト指向の無い Perl 4 のこと[*5] は可及的速やかに忘れてやって下さい。現在絶賛開発中の Perl 6 version 1(?)[*6] については編集部の葡萄酒氏が**素敵な記事**を書いてくれているはず[*7] なのでそちらを参照のこと。

Windows ユーザの方は、手元に Perl の実行環境なんか無いよという人が大多数だと思いますが、その場合は ActiveState から ActivePerl を入手[*8] してきましょう。Windows 上では基本的に

```
D:¥>perl -w hoge.pl
```

---

*4「Perl プログラマのレベル 10」なるものがある(http://d.hatena.ne.jp/naoya/20050809/1123563794)のですが、これでレベル 2 〜 3 と判定される方でも、「大丈夫だ、問題ない。」と言える内容を心がけて書くつもりです。というか私自身もせいぜい 6 〜 7 なので。しかし、「さらにレベル数を大きくしていくと、すぐに該当者が Larry Wall (筆者注:Perl の創始者)だけになります。」というのはあれだな、「本当の Perl 処理系は Larry Wall の頭の中にしか無い」ってやつですね。(違

*5 筆者が Perl に触るようになった 2000 年頃、Web 上で CGI に使われる言語はほとんどが Perl というのが実態でした。もっとも、プロバイダ提供のサーバスペースを含め、レンタルサーバ上で選べる CGI 用の言語が Perl ぐらいしか無かった、という背景がありますが(無料レンタルサーバだと **CGI 機能そのものが無い**ことが多かった)。そして、オブジェクト指向が導入された Perl 5 系列が初めてリリースされた 1994 年からかなり経った 2000 年頃でさえ、CGI 用 Perl プログラムでは「use Hogehoge;」しているものすらほとんどありませんでした。というか、当時の一般的な個人サイトでの CGI プログラムの大半は、掲示板やカウンタなどの比較的簡単な目的しか持たなかったためオブジェクト指向を用いるまでもなく、その結果 Web 上で一番使われる層の Perl のコードは全くオブジェクト指向を用いていないものばかりでした。そのせいか、筆者くらいの年齢層(1980 年代前半生まれ、とより上の世代)だと、Perl でオブジェクト指向プログラミングをすると言うと変な(**なにそれこわい**、的な)顔をされることがままあったのは良い思い出です。

*6 (ほぼ)上位互換だった Perl 4 → 5 とは違い、Perl 5 のコードは基本的に Perl 6 では動かないそうです。伝聞系なのは筆者自身は Perl 6 に全く触れていないから。その代わり、Perl 5 でオブジェクト指向が後付けなせいで色々とコードがフリーダムだったのが 6 ではかなり綺麗な書き方になったとのこと。でも、**綺麗な Perl ってそれってもう Perl じゃない**し、そもそも非互換だからいっそ新しい言語として「Perl 6 version 1」とか呼ぼうぜ、と葡萄酒氏と馬鹿な話をしてました。まぁ、それでも Perl であるからにはきっと**綺麗という言葉の解釈が一般とは違う**という落ちなんだろうけれど。

*7 え?葡萄酒なら私の隣(にあるソファの上)で寝てるよ?
……そんな WORD で大丈夫か?(締め切り的な意味で)

*8 http://www.activestate.com/activeperl/downloads

というような形で Perl インタプリタ[*9] にソースコードを渡して実行することになる[*10] ので、環境変数 PATH に含まれるところであればどこにインストールしても特に問題は無い[*11] でしょう。多くの場合 C:¥Perl または C:¥Perl64[*12] 以下に展開されます。

Windows 以外の環境、特に Linux や MacOSX などの*nix 系はデフォルトで perl がインストールされていることと思います。

筆者は以下の環境でテストしています[*13]。

- ActivePerl 5.12.2 build 1202 x86 および x64 on Windows 7 x64
- perl 5.10.1-8ubuntu2 x64 on Ubuntu Lucid Lynx x64

x86, x64 間では基本的な使い方では差異は無いはずですが、CPAN[*14] からモジュールを持ってくる際[*15] などに問題が起きる場合があるので、それについては適宜記載します。

目次

予定は未定ですが、とりあえず下記を想定しています。

1. use による既存モジュールの活用と package 宣言
2. Perl でのクラス・インスタンス・メソッド
3. 既存モジュールの拡張・継承
4. POE: Perl Object Environment の紹介
5. POE を使ったマルチタスキング
6. POE によるイベントドリブン生活
7. 並列クローラの作成

途中脇道に逸れたりバグに嵌ったり新たな章が追加されたりタイトルが変わったり内容が変わったり**記事が無かったことになったり**するとは思いますが、なるべく完結させるつもり[*16] ですのでどうぞお付き合い下さい。

---

*9 実際には Perl は実行時にコードをコンパイルする JIT コンパイラ形式になっています。

*10 pl2bat という、Perl のソースコードにヘッダとフッタを付け加え Windows バッチファイル形式に変換するコマンドが ActivePerl には付いてくるので、それを利用して*nix における chmod +x のようなことが擬似的に実現できたりはします。あるいは、ActivePerl のデフォルトのインストールオプションでは拡張子 pl のファイルが Perl インタプリタに関連付けされるため、pl ファイルを開くと自動的にインタプリタが起動する環境にすることもできます。

*11 というより、デフォルトのインストールオプションではインストール先を環境変数 PATH に追加してくれます。

*12 x64 版。

*13 2010/10/31 現在。今後 update があればできるだけ追従していきます。

*14 http://www.cpan.org/ "Comprehensive Perl Archive Network"の略。Perl にまつわるライブラリ・モジュールその他いろんなコードの巨大なアーカイブサイトです。方向性はちょっと違いますが ruby でいうと RubyGems とかが近いんですかね。

*15 特に最新のソースを使って **tar.gz から make** したりする場合に。

*16 まぁ、最初から「打ち切り」にするつもりで書き(or 描き)始める作者は少数派でしょうから。…… mitty 先生の次回作にご期待下さい！！！ 1

use Perl::Object;

蛇足

と、本来は本記事は「近日連載開始」的な告知ではなくて内容の伴った記事[*17] にするつもりだった[*18] ですし、このまま終わるのも寂しいので最初のページに載せた mycat.pl の解説をしながら終わりたいと思います。

まず理解してほしいのが、コードには記述されていない特殊変数を Perl は暗黙的に多用する、ということです。具体的には「$_」および今回は出てきませんが、「@_」です[*19]。また、スクリプト起動時に渡されるコマンドライン引数が格納される「@ARGV」も隠れています。これらを省略しない形で mycat.pl を書き直し、また foreach のエイリアスである for を元に戻す[*20] と以下のようになります。

```
1.      #! /usr/bin/perl -w
2.
3.      use strict;
4.      use warnings;
5.
6.      if (! @ARGV) {
7.              @ARGV = ('-');
8.      }
9.      foreach $_ (@ARGV) {
10.             open FileHandle, $_;
11.             foreach $_ (<FileHandle>) {
12.                     print $_;
13.             }
14.     }                                                       # mycat2.pl
```

まず、Perl では制御構文[*21] が後置できます。このため、元の mycat.pl の 6 行目は print 文に for 文を後置したと解釈されます。if 文や unless 文などの条件文の後置くらいだと他の言語でも結構

---

*17 そもそも、元々は目次予定の 4 ～ 5 章あたりをまとめた単体記事にするつもりでした。しかし、「『Perl でオブジェクト指向とか無いわー』とか言われないように入門も付けるか」→「クラスとかの説明もしないといけなくね？」→「分量的に連載になるよな」→「いっそ Perl のオブジェクト指向入門記事ってことにした方が(自分にとっても)勉強になっていい気がする」→（ｒｙ と膨らんだ結果が**ごらんの有様だよ!!!**

*18 締め切り三日前になっていきなり記事を書くことを思い立った時点で間に合わない予感は十二分だったわけですけどね。

*19 ドル記号 or アットマーク＋アンダーバー。これを書いているときに「%_」も暗黙的に定義されていることに気づきましたが、どのように使われているのかが分かりません。「%_」が何に使われているのかご存じな方は是非教えて下さい。

*20 C 言語などの for 文と同じ認識で Perl のループを捉えようとすると**大変なことになりますよ？**

*21 if 文や unless 文、foreach 文(for 文)、while 文、until 文などのこと。

見かけるますが、foreach 文や while 文などの後置[*22] は Perl くらいでしか見かけない気がします。そして、後置した場合のみ、制御構文の条件節の()が省略できます。ここで、ループが一重から二重になってるだろ、とか突っ込みが来そうですが、この程度のことは **Perl では良くあること**なので諦めて下さい[*23]。次に「$_」ですが、Perl では組み込み関数[*24] での引数の省略時やループ構文などでのイテレータの省略時に、デフォルトの対象変数として「$_」を暗黙的に使うというお約束があります。しかも、「$_」に何が入っているかは**その時点での文脈による**[*25] ので、実行時にならないと分からないということが良くあります。

この「$_」が曲者で、Perl のソースコードが Perler[*26] 以外の人にとってスパゲティコードにしか見えない理由の~~大半~~一部[*27] はこいつのせいだと個人的には思っています。逆に Perler が Perl から離れられない理由でもあるでしょうが。

さて、mycat2.pl のコードを一行ずつ見ていきましょう。1 行目は shebang です。3 行目と 4 行目の「use」はモジュールの読み込みでも使われますが、今回のものは C コンパイラでのコンパイルオプションのようなもので、インタプリタの挙動を変更する「プラグマ」と呼ばれています。「use strict;」は文法などが厳格に適用[*28] されるようになり、変数のスコープなどを明確に宣言しないといけなくなります。「use warnings;」は文字通り**怪しい構文**や未定義の変数使用などを警告してくれるようになります。

6 行目以降がコード本体ですが、まず、mycat.pl には存在していなかった if 文が出現している理由を説明しましょう。mycat.pl の 6 行目で使われている「<>」というダイヤモンドみたいなものは「山括弧演算子[*29]」と呼ばれ、ファイルハンドルを引数にしてファイルの内容を一行ずつ読み

---

*22 do {} while 構文や do {} until 構文は、もちろん while 文あるいは until 文の後置とは別に扱われます。というより、Perl では do {} while 構文は制御構文**ではなく**、{}によって囲まれた複数の文を引数とする do 演算子に while 文が後置されていると解釈されます。つまり do 演算子に if 文や unless 文が後置された、do {} if 構文あるいは do {} unless 構文なども正しいコードとなります。また、通常の制御構文である while 文などで使われるループ制御のための next 演算子や last 演算子は、do 演算子が制御構文ではないため、do {} while 構文や do {} until 構文において**使うことができません**。

*23 そういう極端な省略も Perl では普通です。(キリッ

*24 組み込み命令、と捉えた方が使い方の実態に近い気もします。たとえば「print」は関数なので「print "hoge";」ではなく「print("hoge");」という書き方もできますが、あまりやりません。そもそも「print **文**」という呼び方をしますし。

*25 要するに **Perl の気分次第です**。慣れるしかないね。むしろ慣れてくると快感に（ｒｙ

*26 日常的に Perl でコードを書く人たちの総称。

*27 理由の大半、と書こうとしてやめたのは、私くらいでも何とか説明できそうな「$_」の扱われ方程度では **Perl のやばさは全然書き切れない**と思ったからです。

*28 とはいっても、Perl の文法のフリーダムさにしてみれば**不可算無限集合が加算無限集合になった**程度の差しかないと思われます。いや、その差ってかなり大きいですけどね。

*29 あるいは「行入力演算子」とも呼ばれます。実際には「readline」関数を呼んでいるので字義通りの「演算子」ではないのですが、「readline(FileHandle)」と記述することはあまり一般的ではなくたいてい「<FileHandle>」のような書き方をされるので「演算子」と言われた方がしっくり来ますね。

込んで返すという演算子[*30] です。mycat.pl では対象となるファイルハンドルが省略されているため、暗黙的に「open」済みの「ARGV」という特別なファイルハンドルが対象となります。「ARGV」ファイルハンドルは「コマンドラインに指定されたファイル名のリスト@ARGV[*31] を順にオープンして読み込むための特殊ファイルハンドル」[*32] として用意されています。そして、「@ARGV」が空、つまりコマンドライン引数が無い場合、暗黙的に標準入力すなわち「-」（ハイフン）が対象となります。従って、mycat2.pl にあるように if 文による「@ARGV」の判定が必要となります。

9 行目の foreach ループでは、「@ARGV」から要素を一つずつ取り出し「$_」に代入します。そして渡された引数をファイル名文字列として「open」し、「FileHandle[*33]」ハンドラに結びつけます。

11 行目の内側の foreach ループでは、「山括弧演算子」により読み込まれた一行をまたも「$_」に代入し、ループ内の「print」関数で出力しています。

最後の行の行末にある「#」以下の部分はコメントです。

以上が例として挙げた mycat.pl および mycat2.pl の**簡単な**説明です。いかがでしょうか、コードを 10 分の 1 近くまで圧縮できる **Perl のシンプルさ**[*34] を分かっていただけたことと思います。そんな Perl で楽しいオブジェクト指向プログラミングを目指して連載していきたいと思います。

最後に

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。

参考文献

- 「プログラミング Perl 第 3 版 VOLUME 1,2」（オライリー・ジャパン）
- 「PERL HACKS プロが教えるテクニック＆ツール 101 選」（オライリー・ジャパン）
- http://www.perl.org/
- perldoc コマンド

---

*30 一行ずつではなくファイル全体を一気に読み込む構文も書けますが、今回は省略します。

*31「@ARGV」に格納されているコマンドライン引数をファイル名が渡されたと見なすわけです。

*32「プログラミング Perl 第 3 版」 p.780

*33「FileHandle」は特殊変数でもなんでもない、単なるファイルハンドルです。

*34 from http://slashdot.jp/developers/comments.pl?sid=429722&cid=1468207

> *＞シンプルに書かれた Perl*
>
> *同じ処理をするとして*
> *・単純な処理を明快に積み重ねて 20 行で書かれた処理（理解するのに 2 分）*
> *と*
> *・変態じみた正規表現を駆使して 2 行で書かれた処理（理解するのに 20 分）*
> *だと*
> *後者を「シンプル」だと思ってしまうのが Perl 教徒の恐ろしいところ*

*35 http://dev.perl.org/licenses/artistic.html