

Narzędzia informatyczne w językoznawstwie

Perl - Zastowanie modułów

Marcin Junczys-Dowmunt
junczys@amu.edu.pl

Zakład Logiki Stosowanej
<http://www.logic.amu.edu.pl>

12. marca 2008

Co to jest moduł?

- ▶ Moduły dzielą nam program na logiczne części
- ▶ Grupują np. funkcje i zmienne ze sobą powiązane
- ▶ Pozwalają na proste włączanie do naszego programu kodu już wcześniej przez nas napisanego (tego nie będziemy omawiać)
- ▶ Pozwalają na włączanie do programu kodu napisanego przez inne osoby
- ▶ Korzystaliśmy już z modułów `strict` oraz `Data::Dumper`

- ▶ Moduły pragmatyczne, np. `strict`
Informują interpreter Perla np. o trybie działania

- ▶ Moduły pragmatyczne, np. `strict`
Informują interpreter Perla np. o trybie działania
- ▶ Moduły standardowe, np. `Data::Dumper`
Napisane w czystym Perlu

- ▶ Moduły pragmatyczne, np. `strict`
Informują interpreter Perla np. o trybie działania
- ▶ Moduły standardowe, np. `Data::Dumper`
Napisane w czystym Perlu
- ▶ Moduły rozszerzone, np. `XML::Expat`
Napisane np. za pomocą C++, w celu przyspieszenia działania
lub np. interfejsy do systemu operacyjnego

Po co moduły?

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

- ▶ Wiele czynności podstawowych zostało już wcześniej zakodowanych

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

- ▶ Wiele czynności podstawowych zostało już wcześniej zakodowanych
- ▶ Wystarczy znaleźć odpowiedni moduł, przeczytać dokumentację i korzystać z niego do woli

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

- ▶ Wiele czynności podstawowych zostało już wcześniej zakodowanych
- ▶ Wystarczy znaleźć odpowiedni moduł, przeczytać dokumentację i korzystać z niego do woli
- ▶ Zmniejszy to nasz wysiłek, poprawi jakość programu

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

- ▶ Wiele czynności podstawowych zostało już wcześniej zakodowanych
- ▶ Wystarczy znaleźć odpowiedni moduł, przeczytać dokumentację i korzystać z niego do woli
- ▶ Zmniejszy to nasz wysiłek, poprawi jakość programu
- ▶ Możemy znaleźć np. interfejsy do baz danych, parsery XML i HTML, moduły do ściągania stron internetowych, generatory PDF, ...

Prosta odpowiedź: Nie ma sensu powtarzać pracę, którą ktoś przed nami wykonał lepiej!

- ▶ Wiele czynności podstawowych zostało już wcześniej zakodowanych
- ▶ Wystarczy znaleźć odpowiedni moduł, przeczytać dokumentację i korzystać z niego do woli
- ▶ Zmniejszy to nasz wysiłek, poprawi jakość programu
- ▶ Możemy znaleźć np. interfejsy do baz danych, parsery XML i HTML, moduły do ściągania stron internetowych, generatory PDF, ...

Skąd wziąć odpowiednie moduły?

- ▶ Sporo modułów znajduje się standardowo w naszej instalacji, np. `Data::Dumper`, `Encode`, `XML::Parser`, `LWP`, ...

Skąd wziąć odpowiednie moduły?

- ▶ Sporo modułów znajduje się standardowo w naszej instalacji, np. `Data::Dumper`, `Encode`, `XML::Parser`, `LWP`, ...
- ▶ Największym źródłem gotowych modułów jest CPAN - *Comprehensive Perl Archive Network*, ok. 11000 modułów

Skąd wziąć odpowiednie moduły?

- ▶ Sporo modułów znajduje się standardowo w naszej instalacji, np. `Data::Dumper`, `Encode`, `XML::Parser`, `LWP`, ...
- ▶ Największym źródłem gotowych modułów jest CPAN - *Comprehensive Perl Archive Network*, ok. 11000 modułów
- ▶ Problem: Wymaga kompilacji modułów – trudne pod Windowsem

Skąd wziąć odpowiednie moduły?

- ▶ Sporo modułów znajduje się standardowo w naszej instalacji, np. `Data::Dumper`, `Encode`, `XML::Parser`, `LWP`, ...
- ▶ Największym źródłem gotowych modułów jest CPAN - *Comprehensive Perl Archive Network*, ok. 11000 modułów
- ▶ Problem: Wymaga kompilacji modułów – trudne pod Windowsem
- ▶ Rozwiązanie: PPM – *Perl Package Manager*
dostęp do (niezupełniej, ok. 7000 modułów) kopii CPAN dla Active Perl

Skąd wziąć odpowiednie moduły?

- ▶ Sporo modułów znajduje się standardowo w naszej instalacji, np. `Data::Dumper`, `Encode`, `XML::Parser`, `LWP`, ...
- ▶ Największym źródłem gotowych modułów jest CPAN - *Comprehensive Perl Archive Network*, ok. 11000 modułów
- ▶ Problem: Wymaga kompilacji modułów – trudne pod Windowsem
- ▶ Rozwiązanie: PPM – *Perl Package Manager*
dostęp do (niezupełniej, ok. 7000 modułów) kopii CPAN dla Active Perl
- ▶ CPAN pozostaje niezastąpionym źródłem informacji o pakietach

- ▶ Adres: `http://www.cpan.org` lub `http://cpan.perl.org`

- ▶ Adres: `http://www.cpan.org` lub `http://cpan.perl.org`
- ▶ Spis modułów oraz wyszukiwanie: `http://search.cpan.org`

- ▶ Adres: `http://www.cpan.org` lub `http://cpan.perl.org`
- ▶ Spis modułów oraz wyszukiwanie: `http://search.cpan.org`
- ▶ Widzimy, że moduły zostały ułożone w pewnej hierarchii.

- ▶ Adres: <http://www.cpan.org> lub <http://cpan.perl.org>
- ▶ Spis modułów oraz wyszukiwanie: <http://search.cpan.org>
- ▶ Widzimy, że moduły zostały ułożone w pewnej hierarchii.
- ▶ Hierarchia jest raczej tematyczna i niekoniecznie odzwierciedla hierarchię samych modułów

- ▶ Adres: <http://www.cpan.org> lub <http://cpan.perl.org>
- ▶ Spis modułów oraz wyszukiwanie: <http://search.cpan.org>
- ▶ Widzimy, że moduły zostały ułożone w pewnej hierarchii.
- ▶ Hierarchia jest raczej tematyczna i niekoniecznie odzwierciedla hierarchię samych modułów
- ▶ Np. `Chart::Pie` oraz `Chart::Plot` nie są podmodułami jakiegoś tajemniczego modułu `Chart`. Jednak oba moduły służą do tworzenia wykresów (różnych typów, w różne sposoby, pochodzą od różnych autorów)

- ▶ Adres: <http://www.cpan.org> lub <http://cpan.perl.org>
- ▶ Spis modułów oraz wyszukiwanie: <http://search.cpan.org>
- ▶ Widzimy, że moduły zostały ułożone w pewnej hierarchii.
- ▶ Hierarchia jest raczej tematyczna i niekoniecznie odzwierciedla hierarchię samych modułów
- ▶ Np. `Chart::Pie` oraz `Chart::Plot` nie są podmodułami jakiegoś tajemniczego modułu `Chart`. Jednak oba moduły służą do tworzenia wykresów (różnych typów, w różne sposoby, pochodzą od różnych autorów)
- ▶ Proszę poprzeglądać sobie trochę katalog modułów, np. `String Language Text Processing > Lingua::`

Prosty przykład korzystania z modułu

```
1 use strict;

   my $test = {
       tref => [ 1,2,3,4 ]
5  };

   print Dumper($test);
```


Prosty przykład korzystania z modułu

```
1 use strict;  
  
   my $test = {  
       tref => [ 1,2,3,4 ]  
5  };  
  
   print Dumper($test);
```

- ▶ Pojawi się błąd, dlaczego?

Prosty przykład korzystania z modułu

```
1 use strict;  
  
my $test = {  
    tref => [ 1,2,3,4 ]  
5 };  
  
print Dumper($test);
```

- ▶ Pojawi się błąd, dlaczego?
- ▶ Musimy najpierw zaimportować moduł, czyli use Data::Dumper

Prosty przykład korzystania z modułu

```
1 use strict;  
  
my $test = {  
    tref => [ 1,2,3,4 ]  
5 };  
  
print Dumper($test);
```

- ▶ Pojawi się błąd, dlaczego?
- ▶ Musimy najpierw zaimportować moduł, czyli use Data::Dumper
- ▶ Słowo kluczowe use służy do importowania modułów

Prosty przykład korzystania z modułu

```
1 use strict;  
  
my $test = {  
    tref => [ 1,2,3,4 ]  
5 };  
  
print Dumper($test);
```

- ▶ Pojawi się błąd, dlaczego?
- ▶ Musimy najpierw zaimportować moduł, czyli use Data::Dumper
- ▶ Słowo kluczowe use służy do importowania modułów
- ▶ Data::Dumper jest nazwą modułu. Możemy z niej odczytać, że to jest funkcja powiązana ze strukturami danych, jednak nie ma pakietu Data

Inny Przykład: moduł Getopt::Long

```
1 use strict;
  use Getopt::Long;

  my $jezyk = "polski";
5 my $wykrzyk = 0;

  GetOptions(
    "lang=s" => \$jezyk,
    emph => \$wykrzyk,
10 );

  if($jezyk eq "polski") {
    print "Witaj swiecie";
  }
15 elsif($jezyk eq "angielski") {
    print "Hello world";
  }
  print ($wykrzyk) ? "!!!\n" : "\n";
```

- ▶ Nie wszystkie funkcje są od razu dostępne tak jak Dumper

Importowanie funkcji z modułów

- ▶ Nie wszystkie funkcje są od razu dostępne tak jak Dumper
- ▶ Trzeba się wtedy do nich odwołać przez nazwę modułu (przestrzeń nazw)

Importowanie funkcji z modułów

- ▶ Nie wszystkie funkcje są od razu dostępne tak jak Dumper
- ▶ Trzeba się wtedy do nich odwołać przez nazwę modułu (przestrzeń nazw)

```
1 use Encode;  
my $str = "B\x{119}d\x{199} jutro w domu\n";  
my $u_str = Encode::encode_utf8($str);  
my $iso_str = Encode::encode("iso-8859-2", $str);
```


Importowanie funkcji z modułów

- ▶ Nie wszystkie funkcje są od razu dostępne tak jak Dumper
- ▶ Trzeba się wtedy do nich odwołać przez nazwę modułu (przestrzeń nazw)

```
1 use Encode;  
my $str = "B\x{119}d\x{199} jutro w domu\n";  
my $u_str = Encode::encode_utf8($str);  
my $iso_str = Encode::encode("iso-8859-2", $str);
```

- ▶ Drugi sposób to importowanie konkretnej funkcji za pomocą tablicy, która zawiera nazwy potrzebnych funkcji

Importowanie funkcji z modułów

- ▶ Nie wszystkie funkcje są od razu dostępne tak jak Dumper
- ▶ Trzeba się wtedy do nich odwołać przez nazwę modułu (przestrzeń nazw)

```
1 use Encode;  
my $str = "B\x{119}d\x{199} jutro w domu\n";  
my $u_str = Encode::encode_utf8($str);  
my $iso_str = Encode::encode("iso-8859-2", $str);
```

- ▶ Drugi sposób to importowanie konkretnej funkcji za pomocą tablicy, która zawiera nazwy potrzebnych funkcji

```
1 use Encode qw(encode_utf8 encode);  
my $str = "B\x{119}d\x{199} jutro w domu\n";  
my $u_str = encode_utf8($str);  
my $iso_str = encode("iso-8859-2", $str);
```

Gdy nie mamy pożądanego modułu

```
1 use Lingua::Identify;    # niezainstalowany modul
```

Gdy nie mamy pożądanego modułu

```
1 use Lingua::Identify;    # niezainstalowany modul
```

Informacja o błędzie:

```
Can't locate Lingua/Identify.pm in @INC (@INC contains: C:/Perl/site/lib C:/Perl/lib .) at -e line 1
```

- ▶ W konsoli lub w 'Start > Uruchom ...' wpisujemy PPM

Instalacja modułu za pomocą PPM

- ▶ W konsoli lub w 'Start > Uruchom ...' wpisujemy PPM
- ▶ Lub: 'Wszystkie Programy > ActivePerl 5.x > Perl Package Manager'

- ▶ W konsoli lub w 'Start > Uruchom ...' wpisujemy PPM
- ▶ Lub: 'Wszystkie Programy > ActivePerl 5.x > Perl Package Manager'
- ▶ Gdy chcemy zainstalować pakiet 'Lingua::Identify', to szukamy 'Lingua-Identify'

- ▶ W konsoli lub w 'Start > Uruchom ...' wpisujemy PPM
- ▶ Lub: 'Wszystkie Programy > ActivePerl 5.x > Perl Package Manager'
- ▶ Gdy chcemy zainstalować pakiet 'Lingua::Identify', to szukamy 'Lingua-Identify'
- ▶ Wynik zaznaczamy (*Mark for install*) i wykonujemy operację (*Run marked actions*)

- ▶ W konsoli lub w 'Start > Uruchom ...' wpisujemy PPM
- ▶ Lub: 'Wszystkie Programy > ActivePerl 5.x > Perl Package Manager'
- ▶ Gdy chcemy zainstalować pakiet 'Lingua::Identify', to szukamy 'Lingua-Identify'
- ▶ Wynik zaznaczamy (*Mark for install*) i wykonujemy operację (*Run marked actions*)
- ▶ Po zakończeniu instalacji nie powinien już występować komunikat błędu

Przykładowy moduł niestandardowy Lingua::Identify

```
1 use strict;
  use Lingua::Identify qw(langof);
  use Data::Dumper;

5 my $text = "This is an English text";
  my $lang = langof($text);
  print "\"$text\" is in $lang\n";

  my @langs = langof($text);
10 print Dumper(\@langs);

  my %langs = langof($text);
  print Dumper(\%langs);
```