Módulos

SPLN

23 de Março de 2015

Exemplo de um módulo

Ficheiro Fact.m

```
package Fact;
sub fact {
    my $p = 1;
    $p *= $_ for(1..shift);
    $p;
}
1;
```

Ficheiro f.pl

```
use Fact;
print Fact::fact(7), "\n";
```

Exemplo de um módulo

Ficheiro Fact.m

```
package Fact;
sub fact {
    my $p = 1;
    $p *= $_ for(1..shift);
    $p;
}
1;
```

Ficheiro f.pl

```
use Fact;
print fact(7), "\n";
```

Erro

```
Undefined subroutine &main::fact called at x.pl line 2
```

Exemplo de um módulo

Ficheiro Fact.m

```
package Fact;
sub fact {
    my $p = 1;
    $p *= $_ for(1..shift);
    $p;
}
1;
```

Ficheiro f.pl

```
use Fact;
print fact(7), "\n";
```

Erro

```
Undefined subroutine &main::fact called at x.pl line 2
```

Exemplo de um módulo

Ficheiro Fact.m

```
package Fact;
use strict;
use Exporter;
our @ISA = qw(Exporter);
our @EXPORT = qw(fact);
our @EXPORT_OK = qw();

sub fact {
    my $p = 1;
    $p *= $_ for(1..shift);
    $p;
}
1;
```

Ficheiro f.pl

```
use Fact;
print fact(7), "\n";
```

Exemplo de um módulo

Ficheiro Fact.m

```
package Fact;
use strict;
use Exporter;
our @ISA = qw(Exporter);
our @EXPORT = qw();
our @EXPORT_OK = qw(fact);

sub fact {
    my $p = 1;
    $p *= $_ for(1..shift);
    $p;
}
1;
```

Ficheiro f.pl

```
use Fact qw{&fact};
print fact(7), "\n";
```

MeuModulo.pm

```
package MeuModulo;
use strict;
use Exporter;

our $VERSION     = 1.00;
our @ISA         = qw(Exporter);
# O que se exporta automaticamente
our @EXPORT      = ();
# O que se pode exportar
our @EXPORT_OK   = qw(func1 func2);
# Tags que se podem usar no use
our %EXPORT_TAGS = (Primeira =>[&func1],
    Ambas    => [qw(&func1 &func2)]);

sub func1  { return reverse @_  }
sub func2  { return map{ uc }@_ }
sub func3  { print "invisivel" }
1;
```

Anatomia de um módulo

Exemplo de uso

```
use MeuModulo qw(&func1 &func2);
use MeuModulo qw(:Ambas);
```

Objeto simples

```
package Objeto;
sub new {
    my $class = shift;
    return bless {}, $class;
}
1;
```

Um Triângulo

```
package Triangulo;
sub new {
    my ($class, $base, $altura) = @_;
    return bless {b => $base, h => $altura}, $class;
}
sub area {
    my ($self) = @_;
    $self->{b} * $self->{h} / 2;
}
1;
```

Um Triângulo

```
use Triangulo;

my $t = Triangulo->new(2, 3);

print $t->area(), "\n";
```

Rectângulo

```
package Rectangulo;
sub new {
    my ($class, $a, $b) = @_;
    return bless {a => $a, b => $b}, $class;
}
sub area {
    my ($self) = @_;
    $self->{a} * $self->{b};
}
1;
```

Quadrado

```
package Quadrado;
use Rectangulo;
our @ISA = qw(Rectangulo);

sub new {
        my ($class, $a) = @_;
        return Rectangulo::new($class, $a, $a);
}
1;
```

Utilização do Quadrado

```
use Quadrado;
my $q = Quadrado->new(9);
print $q->area(), "\n";
```

Herança múltipla

```
package A;
sub f {"A::f"}
sub g {"A::g"}
package B;
our @ISA=qw(A);
sub g {"B::g"}
package C;
sub f {"C::f"}
sub i {"C::i"}
sub h {"C::h"}
package D;
our @ISA=qw(C);
sub i {shift->SUPER::i() . "D::i"}
package E;
our @ISA=qw(B D);
sub new {return bless {}, shift}
```

Herança múltipla

```
package main;
my $o = E->new();
for my $nome (qw{f g h i}) {
    my $metodo = $o->can($nome);
    print $o->$metodo(), "\n";
}
```

Herança múltipla

```
A::f
B::g
C::h
C::iD::i
```

Herança múltipla

```
package main;
my $o = E->new();
for my $nome (qw{f g h i}) {
    my $metodo = $o->can($nome);
    print $o->$metodo(), "\n";
}
```

Herança múltipla

```
A::f
B::g
C::h
C::iD::i
```

Criar um módulo

```
h2xs -XAn My::New::Module
```

Criar um módulo

1. Documentar primeiro
2. Criar os testes
3. Usar `use strict` e `use warnings`
4. Pensar nos exports
5. Implementar
6. Atualizar o README
7. Colocar todos os ficheiros no MANIFEST
8. Colocar os prerequisitos no Makefile.PL
9. Atualizar os números de versão

Documentação e Testes

- http://perldoc.perl.org/perlpod.html
- http://perldoc.perl.org/Test/Simple.html
- http://perldoc.perl.org/Test/More.html
- http://perldoc.perl.org/Test/Harness.html