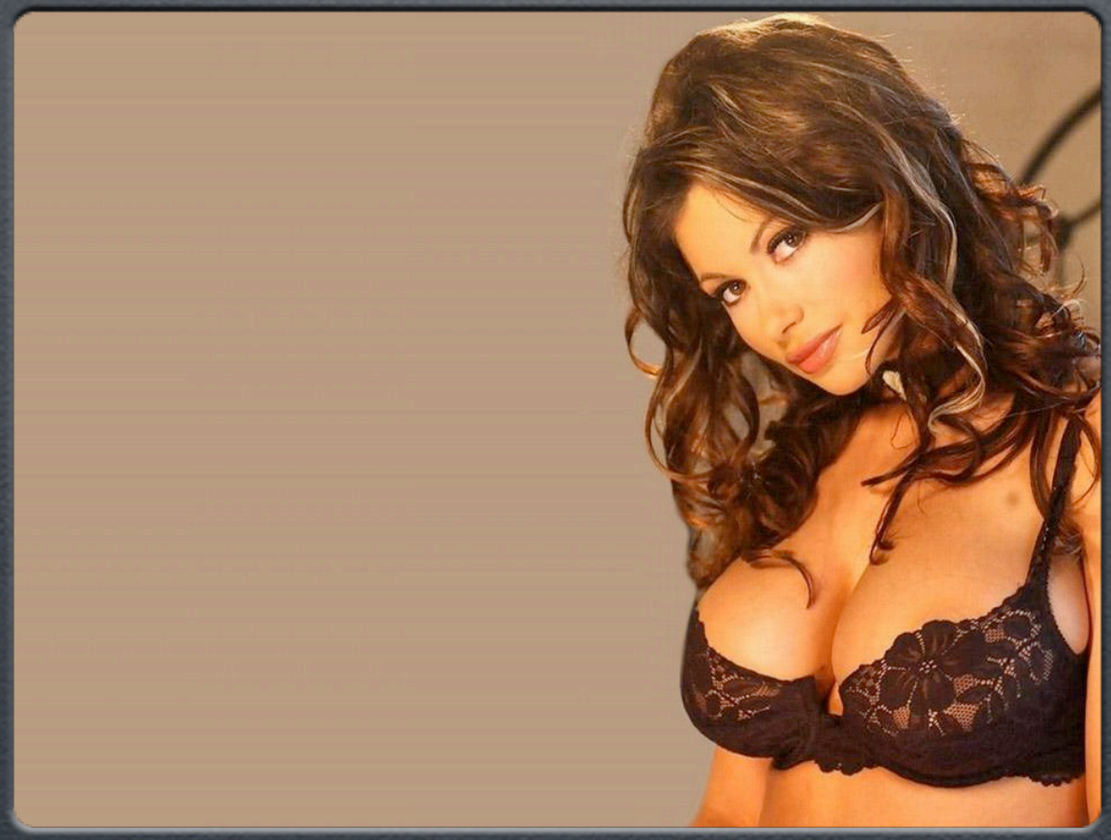
Atributs

Xavier Noria
Barcelona.pm

Abril de 2006



... a Perl

Subrutines

To declare subroutines:

```
    sub NAME;                       # A "forward" declaration.
    sub NAME(PROTO);                #  ditto, but with prototypes
    sub NAME : ATTRS;               #  with attributes
    sub NAME(PROTO) : ATTRS;        #  with attributes and prototypes

    sub NAME BLOCK                  # A declaration and a definition.
    sub NAME(PROTO) BLOCK           #  ditto, but with prototypes
    sub NAME : ATTRS BLOCK          #  with attributes
    sub NAME(PROTO) : ATTRS BLOCK #  with prototypes and attributes
```

To define an anonymous subroutine at runtime:

```
    $subref = sub BLOCK;                    # no proto
    $subref = sub (PROTO) BLOCK;            # with proto
    $subref = sub : ATTRS BLOCK;            # with attributes
    $subref = sub (PROTO) : ATTRS BLOCK; # with proto and attributes
```

Catalyst

```
package MyApp::Controller::Base;

sub authenticate :Private {
    my ($self, $c) = @_;
    # ...
}
```

Variables

```
our EXPR
our EXPR TYPE
our EXPR : ATTRS
our TYPE EXPR : ATTRS
```

Tie::Hash::Regex

```
use Tie::Hash::Regex;

my %h : Regex;

$h{key}   = 'value';
$h{key2}  = 'another value';
$h{stuff} = 'something else';

print $h{key};  # value
print $h{2};    # another value
print $h{'^s'}; # something else
```

Warning

- La semàntica i interfície dels atributs de variables està en evolució i pot canviar
- Veure "Private Variables via my()" a *perlsub* i la documentació d'*attributes.pm*

Sintaxi

- Es poden associar **un o més atributs**
- Aquests se separen per **espais o “:”**
- Els noms han de ser **identificadors simples**
- Poden estar seguits de “(***coses***)”

Ús i attributes.pm

- Si la declaració d'una variable o subrutina té atributs associats Perl li passa informació de context al pragma *attributes.pm*
- No cal carregar *attributes.pm* llevat que es vulgui usar alguna de les seves subrutines
- Els atributs de subrutines i variables `our()` es processen en temps de compilació, els de variables `my()` en temps d'execució

Atributs predefinits

- Hi ha alguns atributs de subrutines ja definits
  - `locked`
  - `method`
  - `lvalue`
- I un per a variables `our()`
  - `unique`

Definició d'atributs

- Cal implementar una subrutina anomenada `MODIFY_`*type*`_ATTRIBUTES`, on *type* és `SCALAR`, `ARRAY`, `HASH`, o `CODE`
- Aquestes funcions reben dos arguments fixos, nom de paquet i referència a la subrutina o variable, seguits de la llista d'atributs
- Retornen la llista d'atributs no reconeguts, que eventualment és la llista buida

Visibilitat

- Un cop definits, els atributs poden usar-se en el package on `MODIFY_`*type*`_ATTRIBUTES` està implementada o derivats
- En particular es pot definir la subrutina com `Universal::MODIFY_`*type*`_ATTRIBUTES` per a que no calgui derivar explícitament

Exemple

```
sub MODIFY_CODE_ATTRIBUTES {
    my ($pkg, $ref, @attrs) = @_;
    print <<EOS;
pkg   -> $pkg
ref   -> $ref
attrs -> @attrs
EOS
    return;
}
```

Catalyst

```
# Veure lib/Catalyst/AttrContainer.pm
sub MODIFY_CODE_ATTRIBUTES {
    my ( $class, $code, @attrs ) = @_;
    $class->_attr_cache( { %{ $class->_attr_cache }, $code => [@attrs] } );
    $class->_action_cache(
        [ @{ $class->_action_cache }, [ $code, [@attrs] ] ] );
    return ();
}
```

Attribute::Handlers

- Abstracció sobre *attributes.pm*
- Procés de la llista, llevat de `RAWDATA`
- `ANY`
- `BEGIN`, `CHECK`, `INIT`, `END`
- Autotie

Exemple

```
package Barcelona::PM::TraceDeclaration;
use Attribute::Handlers;

sub TraceDeclaration :ATTR(CODE) {
    my ($pkg, $symbol, $ref, $attr, $data, $phase) = @_;
    print <<EOS;
paquet al qual pertany la subrutina  -> $pkg
typeglob que conté la subrutina      -> $symbol
referència a la subrutina            -> $ref
nom de l'atribut                     -> $attr
contingut de la llista associada     -> $data
fase en la qual estem essent cridats -> $phase
EOS
}

1;
```

Casos frontera

- El paquet en variables `my()` és `'LEXICAL'`
- Els handlers en variables `my()` no es criden fins runtime, per tant `BEGIN`, `CHECK`, `INIT` no tenen efecte (prova i error)
- L'entrada a la taula de símbols d'una subrutina anònima és `'ANON'`

- L'entrada a la taula de símbols que rep un handler de tipus `CODE` executat en fase `BEGIN` és també `'ANON'` (prova i error)
- Si no hi ha llista associada `$data` és la cadena buida (prova i error)

Ús

- Heretem d'*Attribute::Handlers* (sembla que `use()` és suficient i així està a la SYNOPSIS, però la documentació pròpia no ho diu)
- Definim handlers com a subrutines
  - Mateix nom que l'atribut
  - Atribut `ATTR`, `ATTR(HASH, END)`, etc.
- Els atributs poden usar-se en el package on s'han definit els handlers o derivats

Exemple

```
use Attribute::Handlers;
use Tie::Cycle;

sub Cycle :ATTR(SCALAR) {
    my ($package, $symbol, $referent, $attr, $data, $phase) = @_;
    $data = [ $data ] unless ref $data eq 'ARRAY';
    tie $$referent, 'Tie::Cycle', $data;
}

my $next :Cycle('A' .. 'Z');
print "$next $next $next"; # A B C
```

Autotie

```
use Attribute::Handlers autotie => { Cycle => Tie::Cycle };

my $next :Cycle(['A' .. 'Z']);
print "$next $next $next"; # A B C
```

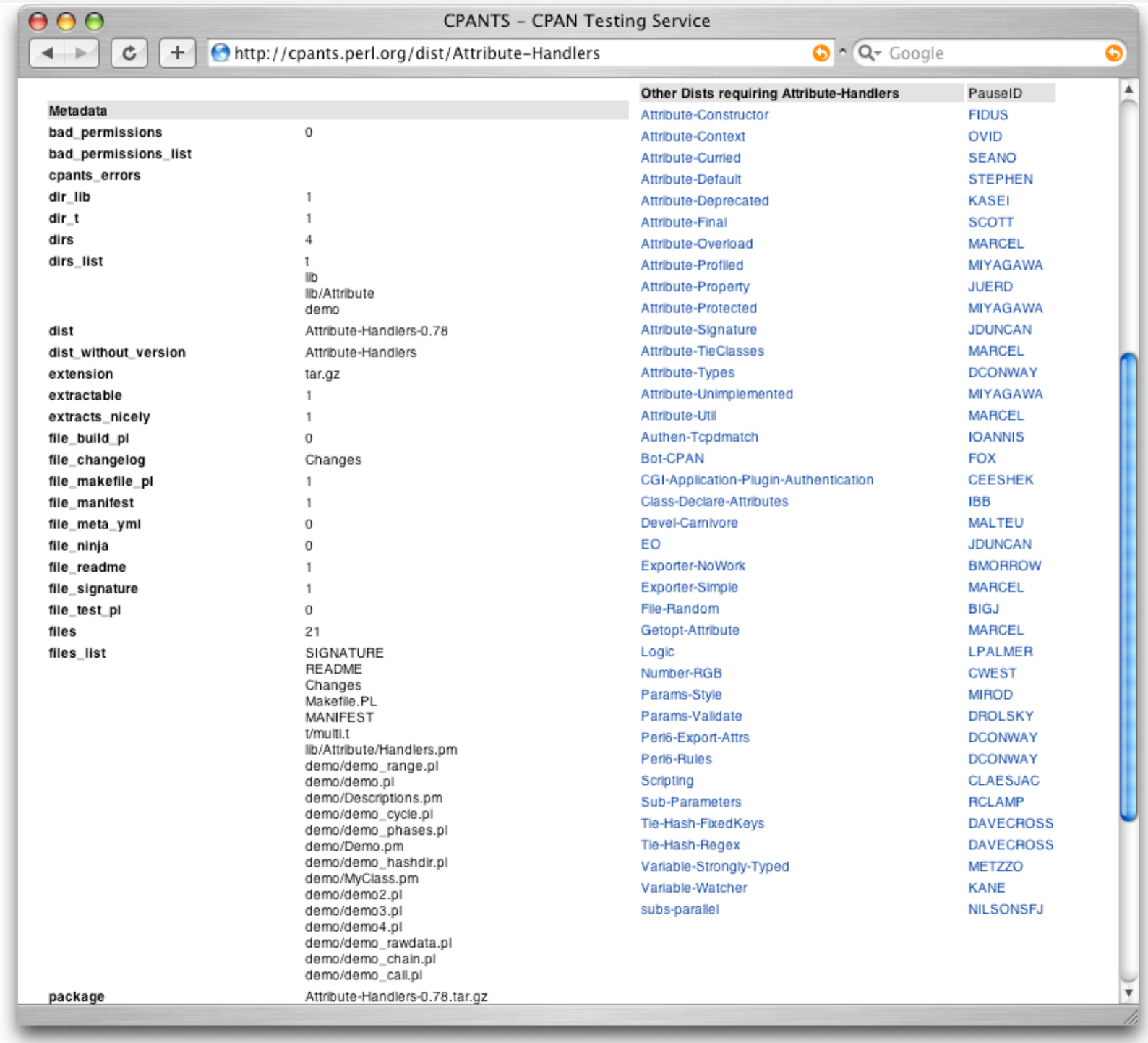
Attribute::Params::Validate

```
sub UNIVERSAL::Validate : ATTR(CODE, INIT)
{
    _wrap_sub('named', @_);
}

sub UNIVERSAL::ValidatePos : ATTR(CODE, INIT)
{
    _wrap_sub('positional', @_);
}

# Només he deixat coses a destacar
sub _wrap_sub
{
    my ($type, $package, $symbol, $referent, $attr, $params) = @_;
    << process stuff >>
    my $subname = $package . '::' . *{$symbol}{NAME};
    my $sub = << define a wrapped subroutine >>;
    *{$subname} = $sub;
}
```

Attribute::Handlers a CPAN



Metadata	
bad_permissions	0
bad_permissions_list	
cpants_errors	
dir_lib	1
dir_t	1
dirs	4
dirs_list	t lib lib/Attribute demo
dist	Attribute-Handlers-0.78
dist_without_version	Attribute-Handlers
extension	tar.gz
extractable	1
extracts_nicely	1
file_build_pl	0
file_changelog	Changes
file_makefile_pl	1
file_manifest	1
file_meta_yml	0
file_ninja	0
file_readme	1
file_signature	1
file_test_pl	0
files	21
files_list	SIGNATURE README Changes Makefile.PL MANIFEST t/multi.t lib/Attribute/Handlers.pm demo/demo_range.pl demo/demo.pl demo/Descriptions.pm demo/demo_cycle.pl demo/demo_phases.pl demo/Demo.pm demo/demo_hashdir.pl demo/MyClass.pm demo/demo2.pl demo/demo3.pl demo/demo4.pl demo/demo_rawdata.pl demo/demo_chain.pl demo/demo_call.pl
package	Attribute-Handlers-0.78.tar.gz

Other Dists requiring Attribute-Handlers	PauseID
Attribute-Constructor	FIDUS
Attribute-Context	OVID
Attribute-Curried	SEANO
Attribute-Default	STEPHEN
Attribute-Deprecated	KASEI
Attribute-Final	SCOTT
Attribute-Overload	MARCEL
Attribute-Profiled	MIYAGAWA
Attribute-Property	JUERD
Attribute-Protected	MIYAGAWA
Attribute-Signature	JDUNCAN
Attribute-TieClasses	MARCEL
Attribute-Types	DCONWAY
Attribute-Unimplemented	MIYAGAWA
Attribute-Util	MARCEL
Authen-Tcpdmatch	IOANNIS
Bot-CPAN	FOX
CGI-Application-Plugin-Authentication	CEESHEK
Class-Declare-Attributes	IBB
Devel-Carnivore	MALTEU
EO	JDUNCAN
Exporter-NoWork	BMORROW
Exporter-Simple	MARCEL
File-Random	BIGJ
Getopt-Attribute	MARCEL
Logic	LPALMER
Number-RGB	CWEST
Params-Style	MIROD
Params-Validate	DROLSKY
Perl6-Export-Attrs	DCONWAY
Perl6-Rules	DCONWAY
Scripting	CLAESJAC
Sub-Parameters	RCLAMP
Tie-Hash-FixedKeys	DAVECROSS
Tie-Hash-Regex	DAVECROSS
Variable-Strongly-Typed	METZZO
Variable-Watcher	KANE
subs-parallel	NILSONSFJ

Gràcies!