# Non-canonical XS Objects With a Bit of Perl Magic

Sergey Aleynikov

YAPC::Russia 2015

Crazy Panda

## Canonical XS objects

```
MODULE = DateTime   PACKAGE = DateTime

SV*
new(const char* CLASS)
CODE:
    DateTime* THIS = new DateTime();

    RETVAL = newRV_noinc(newSViv(PTR2IV(THIS)));
    sv_bless(RETVAL, gv_stashpv(CLASS, 0));
OUTPUT:
    RETVAL
```

## Accessing canonical object

```
MODULE = DateTime   PACKAGE = DateTime

void
dump(SV* obj)
CODE:
    if (!sv_isobject(obj))
        croak("Not a DateTime object");

    DateTime* THIS = (DateTime*)SvIV(SvRV(obj));
    THIS->dump;
```

## Write less with typemaps

```
typemap
DateTime* O_OBJECT

DateTime.xs
MODULE = DateTime   PACKAGE = DateTime

DateTime*
DateTime::new()
CODE:
    RETVAL = new DateTime();
OUTPUT:
    RETVAL

void
DateTime::dump()
CODE:
    THIS->dump();
```

## Why not?

**Pros**

- Straightforward
- Inside a default typemap
- Fast unpack

## Why not?

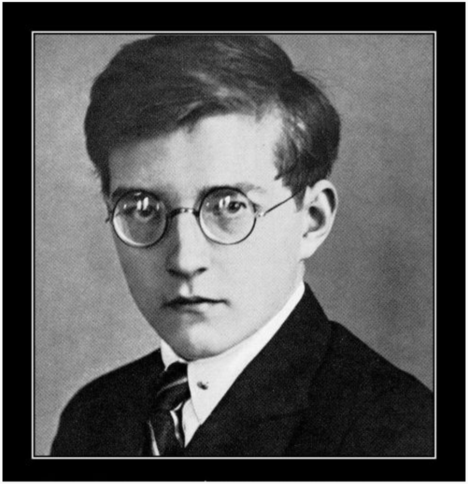
**Pros**

- Straightforward
- Inside a default typemap
- Fast unpack

**Cons**

- No additional data
- One C object per Perl object
- Visible at Perl level

## What is Perl magic?



```
perl -e ’$??s:;s:s;;$?::s;;=]=>%-{<-|}<&|‘{;;y; -/:-@[-‘{-};‘-{/" -;;s;;$_;see’
```

## What is Perl magic indeed?

## True magic

- @ISA
- %^H
- %SIG
- $!
- $DB::single
- referee behind weaken()'d reference
- tie()'d variable
- ...more than 40 types

## How it works?

- Acts on event trigger
    - svt_get
    - svt_set
    - svt_free
    - svt_clear

## How it works?

- Acts on event trigger
    - svt_get
    - svt_set
    - svt_free
    - svt_clear
- Special types are checked at various places

## How it works?

- Acts on event trigger
    - svt_get
    - svt_set
    - svt_free
    - svt_clear
- Special types are checked at various places
- PERL_MAGIC_ext reserved for extensions
- sv_magicext API call

## Creating magical object

```
STATIC MGVTL marker;

MODULE = DateTime   PACKAGE = DateTIme

SV*
new(const char* CLASS)
CODE:
    SV* obj = newHV();
    DateTime THIS* = new DateTIme();
    sv_magicext(obj, NULL, PERL_MAGIC_ext, &marker,
        (const char*)THIS, 0);
    SvRMAGICAL_off(obj);

    RETVAL = newRV_noinc(obj);
    sv_bless(RETVAL, gv_stashpv(CLASS, 0));
OUTPUT:
    RETVAL
```

## Accessing magical object

```
MODULE = DateTime   PACKAGE = DateTIme

void
dump(SV* obj)
CODE:
    if (!SvROK(obj)) croak("Not a DateTime object");

    MAGIC* mg = mg_findext(SvRV(obj), PERL_MAGIC_ext, &marker);
    if (!mg) croak("Not a DateTime object");

    DateTime THIS* = (DateTime*)(mg->mg_ptr);
    THIS->dump();
```

## Lifecycle

```
MODULE = DateTime   PACKAGE = DateTIme

void
DESTROY(SV* obj)
CODE:
    MAGIC* mg = mg_findext(SvRV(obj), PERL_MAGIC_ext, &marker);
    if (mg) {
        DateTime THIS* = (DateTime*)(mg->mg_ptr);
        delete THIS;
    }
```

## Example

```perl
use DateTime;
use DDP;

my $foo = DateTime->new;
$foo->{bar} = 42;

$f->dump;
p $foo;
```

## Example

```
use DateTime;
use DDP;

my $foo = DateTime->new;
$foo->{bar} = 42;

$f->dump;
p $foo;
```

```
% perl test.pl
dump
DateTime    {
    internals:  {
        bar 42
    }
}
```

## Extending objects

- Add data & methods to pointer-based objects
- Add data to perl hashes
- Attach multiple C++ objects
- Attach Perl data to CV*

## Closures?

```
__PACKAGE__->install_accessor("foo", $bar);
__PACKAGE__->install_accessor("bar", $baz);

sub install_accessor {
    my ($package, $name, $data) = @_;

    no strict 'refs';
    *{$package.'::'.$name} = sub {
        return $data;
    }
}
```

## Real magic

```
CV*
Perl_newXS_flags(pTHX_ const char* name, XSUBADDR_t subaddr,
    const char* const filename, const char* proto, I32 flags);

void
install_accessor(pTHX_ const char* name, SV* data) {
    CV* cv = newXS_flags(name, xs_accessor, __FILE__, NULL, 0);

#ifndef MULTIPLICITY
    CvXSSUBANY(cv).any_ptr = (void*)data;
#endif

    sv_magicext((SV*)cv, data, PERL_MAGIC_ext, &marker, NULL, 0);
    SvREFCNT_dec_NN(data);
    SvRMAGICAL_off((SV*)cv);
}
```

Questions?

github://Class::Accessor::Inherited::XS

cpan://Panda::XS