How can I determine if a Perl function exists at runtime?

Asked 15 years, 11 months ago Modified 1 year, 11 months ago Viewed 22k times

26

I'm working on a test framework in Perl. As part of the tests, I may need to add precondition or postcondition checks for any given test, but not necessarily for all of them. What I've got so far is something like:

```
eval "&verify_precondition_TEST$n";
print $@ if $@;
```

Unfortunately, this outputs "Undefined subroutine &verify_precondition_TEST1 called at ..." if the function does not exist.

How can I determine ahead of time whether the function exists, before trying to call it?

perl introspection

Share
Improve this question
Follow

edited Jan 12, 2009 at 18:52

brian d foy
132k ● 31 ● 211 ● 604

asked Jan 11, 2009 at 21:28

Greg Hewgill
990k ● 191 ● 1.2k ● 1.3k

4 Answers

Sorted by: Highest score (default)

43

```
Package::Name->can('function')
```

or

```
*Package::Name::function{CODE}

# or no strict; *{ "Package::Name::$function" }{CODE}
```

or just live with the exception. If you call the function in an eval and $@ is set, then you can't call the function.

Finally, it sounds like you may want Test::Class instead of writing this yourself.

Edit: `defined &function_name` (or the `no strict; defined &{ $function_name }` variant), as mentioned in the other answers, looks to be the best way. UNIVERSAL::can is best for something you're going to call as a method (stylistically), and why bother messing around with the symbol table when Perl gives you syntax to do what you want.

Learning++ :)

Share

Improve this answer

Follow

edited Jan 12, 2009 at 1:52

answered Jan 11, 2009 at 21:43

jrockway

42.6k ● 9 ● 64 ● 87

Unfortunately I'm doing this outside any package, so I get: Can't locate object method "can" via package "main". Also, I'm using an ancient version of Perl (5.002) in an environment where I have absolutely no Perl modules installed. – Greg Hewgill Jan 11, 2009 at 22:18

Whoa, 5.002? I think that came out before I was born :) – jrockway Jan 11, 2009 at 22:30

Isn't 5.002 the one with all of the buffer overflow problems? :) – brian d foy Jan 12, 2009 at 18:55

According to perlhist, 5.003 is 5.002 with security fixes. – jrockway Jan 12, 2009 at 19:57

(And, they didn't have UNIVERSAL:: back then, apparently. Which is why main->can doesn't work; it works fine under 5.10.) – jrockway Jan 12, 2009 at 19:58

20

```
sub function_exists {
    no strict 'refs';
    my $funcname = shift;
    return \&{$funcname} if defined &{$funcname};
    return;
}

if (my $subref = function_exists("verify_precondition_TEST$n") {
    ...
}
```

Share

Improve this answer

Follow

edited Jan 13, 2009 at 16:00

answered Jan 12, 2009 at 1:21

Leon Timmermans

30.2k ● 2 ● 64 ● 110

1 +1 (no strict is always better than string eval) BTW, tiny typo: s/$functname/$funcname/ – jrockway Jan 12, 2009 at 1:53

I'd write that as `no strict 'refs'` just to note to the reader that you're about to do some symbolic reference magic. :) – brian d foy Jan 12, 2009 at 18:54

10

With defined:

```
if (eval "defined(&verify_precondition_TEST$n)") {
    eval "&verify_precondition_TEST$n";
    print $@ if $@;
}
else {
    print "verify_precondition_TEST$n does not exist\n";
}
```

EDIT: hmm, I only thought of eval as it was in the question but with symbolic references brought up with Leon Timmermans, couldn't you do

```
if (defined(&{"verify_precondition_TEST$n"}) {
    &{"verify_precondition_TEST$n"};
    print $@ if $@;
}
else {
    print "verify_precondition_TEST$n does not exist\n";
}
```

even with strict?

Share

Improve this answer

Follow

edited Apr 26, 2012 at 22:01

Paul Tomblin

183k ● 59 ● 323 ● 410

answered Jan 11, 2009 at 21:49

Timo Metsälä

366 ● 1 ● 5

1 This is the wrong eval. You want block eval, not string eval. Plus testing if a function exists can be done with an eval anyway. – Dave Rolsky Jan 11, 2009 at 22:51

I need to use string eval because $n is not known at compile time. – Greg Hewgill Jan 11, 2009 at 22:59

No you don't. You need to use symbolic references. See my answer for an example of how to approach them. – Leon Timmermans Jan 12, 2009 at 1:24

You're right, defined is exempt from strict refs. However, calling a sub isn't. You have to either do 'no strict' or use an eval, the former being cleaner IMNSHO – Leon Timmermans Jan 12, 2009 at 16:37

Also, you need to use package name if the function is defined in some package... for example `use CGI; my $q=CGI->new; my @users = defined (&CGI::multi_param) ? $q->multi_param('u') : $q->param('u');` – Matija Nalis Apr 3, 2018 at 13:32

5

I had used Leon's approach, but when I had multiple packages, it failed. I'm not sure precisely why; I think it relates to the propagation of scope between namespaces. This is the solution I came up with.

```
my %symbols = ();
my $package = __PACKAGE__; # bring it in at run-time
{
    no strict;
    %symbols = %{$package . "::"}; #S ee Symbol Tables on perlmod
}
print "$funcname not defined\n" if (!defined($symbols{$funcname});
```

References:
__PACKAGE__ reference on the perlmod page.

Packages/__PACKAGE__reference on Perl Training Australia.

Share

Improve this answer

Follow

edited Jan 18, 2023 at 11:22

pevik
4,781 ● 3 ● 36 ● 49

answered Aug 4, 2009 at 17:58

Paul Nathan
40.2k ● 30 ● 120 ● 215

2 The problem might be if you've tried to do directly like: `if ( defined &{$package . '::subname'} )` , but like this is should be fine: `my $subname = $package.'::sub_name'; if ( defined &{$subname} ) ...` – bor Sep 5, 2014 at 10:24