'Only'

Managing multiple versions of modules
Installation

Makefile.PL style

cd MyModule-0.30
perl Makefile.PL
make test
perl -Monly=install  # substitute for 'make install'
perl -Monly=install - version=0.33 versionlib=/home/tony/perlmods

Installing with Module::Build

When installing modules distributed with Module::Build, you can use the following commands to install into version specific libraries:

perl Build.PL
./Build
./Build versioninstall

For overrides:

perl Build.PL version=1.23 versionlib=/home/ingy/modules
./Build
./Build versioninstall
Usage

use only MODULE => CONDITION;
use only MODULE => CONDITION, ARGUMENTS;
use only MODULE =>
[ CONDITION1, ARGUMENTS1 ],
..., 
[ CONDITION7, ARGUMENTS7 ];
use only { versionlib => '<version lib>' };
use only { versionlib => '<version lib>' } MODULE => CONDITION, ARGUMENTS;

# Object Oriented Interface
use only;
my $only = only->new;
$only->module('<Module>');
$only->condition('<Version>');
$only->include;
use MyModule;
$only->remove;
Examples

use only MyModule => '0.11';
use only MyModule => '0.11 0.15';
use only MyModule => '0.11-0.12';
use only MyModule => '0.13-';
use only MyModule => '-0.10';
use only MyModule => '-';  # Means any version
use only MyModule => '!0.10'; # Any version but 0.10
use only Inline => 0.44, 'Java'; # passing arguments

use only;
my $only;
BEGIN {
    $only = only->new->module('Good::Stuff')->condition('1.20-1.55')->include;
}
use Good::Stuff;
print "Using Good::Stuff version: " . $only->distribution_version . "\n";
...
$only->remove;  # Remove object from @INC;
Why use Only?

- Need to upgrade a module in a production environment when other systems are using the installed Perl.
- Two versions of a Module are needed at the same time (e.g. DBD::Sybase).
- Safer module upgrade route - it is difficult to downgrade a module once upgraded - install upgraded module with only - and test it before upgrading the core module.