NAME

CPAN::Meta::Requirements - a set of version requirements for a CPAN dist

VERSION

version 2.125

SYNOPSIS

```
use CPAN::Meta::Requirements;

my $build_requires = CPAN::Meta::Requirements->new;

$build_requires->add_minimum('Library::Foo' => 1.208);

$build_requires->add_minimum('Library::Foo' => 2.602);

$build_requires->add_minimum('Module::Bar' => 'v1.2.3');

$METAyml->{build_requires} = $build_requires->as_string_hash;
```

DESCRIPTION

A CPAN::Meta::Requirements object models a set of version constraints like those specified in the *META.yml* or *META.json* files in CPAN distributions. It can be built up by adding more and more constraints, and it will reduce them to the simplest representation.

Logically impossible constraints will be identified immediately by thrown exceptions.

METHODS

new

```
my $req = CPAN::Meta::Requirements->new;
```

This returns a new CPAN::Meta::Requirements object. It takes an optional hash reference argument. The following keys are supported:

- <bad_version_hook> — if provided, when a version cannot be parsed into

  a version object, this code reference will be called with the invalid version string as an argument. It must return a valid version object.

All other keys are ignored.

add_minimum

```
$req->add_minimum( $module => $version );
```

This adds a new minimum version requirement. If the new requirement is redundant to the existing specification, this has no effect.

Minimum requirements are inclusive. `$version` is required, along with any greater version number.

This method returns the requirements object.

add_maximum

```
$req->add_maximum( $module => $version );
```

This adds a new maximum version requirement. If the new requirement is redundant to the existing specification, this has no effect.

Maximum requirements are inclusive. No version strictly greater than the given version is allowed.

This method returns the requirements object.

add_exclusion

```
$req->add_exclusion( $module => $version );
```

This adds a new excluded version. For example, you might use these three method calls:

```
$req->add_minimum( $module => '1.00' );
$req->add_maximum( $module => '1.82' );

$req->add_exclusion( $module => '1.75' );
```

Any version between 1.00 and 1.82 inclusive would be acceptable, except for 1.75.

This method returns the requirements object.

exact_version

```
$req->exact_version( $module => $version );
```

This sets the version required for the given module to *exactly* the given version. No other version would be considered acceptable.

This method returns the requirements object.

add_requirements

```
$req->add_requirements( $another_req_object );
```

This method adds all the requirements in the given CPAN::Meta::Requirements object to the requirements object on which it was called. If there are any conflicts, an exception is thrown.

This method returns the requirements object.

accepts_module

```
my $bool = $req->accepts_modules($module => $version);
```

Given an module and version, this method returns true if the version specification for the module accepts the provided version. In other words, given:

```
Module => '>= 1.00, < 2.00'
```

We will accept 1.00 and 1.75 but not 0.50 or 2.00.

For modules that do not appear in the requirements, this method will return true.

clear_requirement

```
$req->clear_requirement( $module );
```

This removes the requirement for a given module from the object.

This method returns the requirements object.

requirements_for_module

```
$req->requirements_for_module( $module );
```

This returns a string containing the version requirements for a given module in the format described in CPAN::Meta::Spec or undef if the given module has no requirements. This should only be used for informational purposes such as error messages and should not be interpreted or used for comparison (see ‘‘accepts_module’’ instead.)

required_modules

This method returns a list of all the modules for which requirements have been specified.

clone

```
$req->clone;
```

This method returns a clone of the invocant. The clone and the original object can then be changed independent of one another.

is_simple

This method returns true if and only if all requirements are inclusive minimums — that is, if their string expression is just the version number.

is_finalized

This method returns true if the requirements have been finalized by having the `finalize` method called on them.

finalize

This method marks the requirements finalized. Subsequent attempts to change the requirements will be fatal, *if* they would result in a change. If they would not alter the requirements, they have no effect.

If a finalized set of requirements is cloned, the cloned requirements are not also finalized.

as_string_hash

This returns a reference to a hash describing the requirements using the strings in the *META.yml* specification.

For example after the following program:

```
my $req = CPAN::Meta::Requirements->new;

$req->add_minimum('CPAN::Meta::Requirements' => 0.102);

$req->add_minimum('Library::Foo' => 1.208);

$req->add_maximum('Library::Foo' => 2.602);

$req->add_minimum('Module::Bar' => 'v1.2.3');

$req->add_exclusion('Module::Bar' => 'v1.2.8');

$req->exact_version('Xyzzy' => '6.01');

my $hashref = $req->as_string_hash;
```

`$hashref` would contain:

```
{
'CPAN::Meta::Requirements' => '0.102',
'Library::Foo' => '>= 1.208, <= 2.206',
'Module::Bar' => '>= v1.2.3, != v1.2.8',
'Xyzzy' => '== 6.01',
}
```

add_string_requirement

```
$req->add_string_requirement('Library::Foo' => '>= 1.208, <= 2.206');
```

This method parses the passed in string and adds the appropriate requirement for the given module. It understands version ranges as described in the ‘‘Version Ranges’’ in CPAN::Meta::Spec. For example:

1.3
>= 1.3
<= 1.3
== 1.3
!= 1.3
> 1.3
< 1.3
>= 1.3, != 1.5, <= 2.0

A version number without an operator is equivalent to specifying a minimum (`>=`). Extra whitespace is allowed.

from_string_hash

```
my $req = CPAN::Meta::Requirements->from_string_hash( \%hash );
```

This is an alternate constructor for a CPAN::Meta::Requirements object. It takes a hash of module names and version requirement strings and returns a new CPAN::Meta::Requirements object.

SUPPORT

Bugs / Feature Requests

Please report any bugs or feature requests through the issue tracker at <https://github.com/dagolden/CPAN-Meta-Requirements/issues>. You will be notified automatically of any progress on your issue.

Source Code

This is open source software. The code repository is available for public review and contribution under the terms of the license.

<https://github.com/dagolden/CPAN-Meta-Requirements>

```
git clone https://github.com/dagolden/CPAN-Meta-Requirements.git
```

AUTHORS

- David Golden <dagolden@cpan.org>
- Ricardo Signes <rjbs@cpan.org>

COPYRIGHT AND LICENSE