NAME

CPANPLUS::Dist::Build

SYNOPSIS

```
    my $build = CPANPLUS::Dist->new(
                                format  => 'CPANPLUS::Dist::Build',
                                module  => $modobj,
                            );

    $build->prepare;    # runs Module::Build->new_from_context;
    $build->create;     # runs build && build test
    $build->install;    # runs build install
```

DESCRIPTION

`CPANPLUS::Dist::Build` is a distribution class for `Module::Build` related modules. Using this package, you can create, install and uninstall perl modules. It inherits from `CPANPLUS::Dist`.

Normal users won't have to worry about the interface to this module, as it functions transparently as a plug-in to `CPANPLUS` and will just `Do The Right Thing` when it's loaded.

ACCESSORS

parent()

Returns the `CPANPLUS::Module` object that parented this object.

status()

Returns the `Object::Accessor` object that keeps the status for this module.

STATUS ACCESSORS

All accessors can be accessed as follows: $build->status->ACCESSOR

build_pl ()

Location of the Build file. Set to 0 explicitly if something went wrong.

build ()

BOOL indicating if the `Build` command was successful.

test ()

BOOL indicating if the `Build test` command was successful.

prepared ()

BOOL indicating if the `prepare` call exited succesfully This gets set after `perl Build.PL`

distdir ()

Full path to the directory in which the `prepare` call took place, set after a call to `prepare`.

created ()

BOOL indicating if the `create` call exited succesfully. This gets set after `Build` and `Build test`.

installed ()

BOOL indicating if the module was installed. This gets set after `Build install` exits successfully.

uninstalled ()

BOOL indicating if the module was uninstalled properly.

_create_args ()

Storage of the arguments passed to `create` for this object. Used for recursive calls when satisfying prerequisites.

_install_args ()

Storage of the arguments passed to `install` for this object. Used for recursive calls when satisfying prerequisites.

_mb_object ()

Storage of the `Module::Build` object we used for this installation.

METHODS

$bool = CPANPLUS::Dist::Build->format_available();

Returns a boolean indicating whether or not you can use this package to create and install modules in your environment.

$bool = $dist->init();

Sets up the `CPANPLUS::Dist::Build` object for use. Effectively creates all the needed status accessors.

Called automatically whenever you create a new `CPANPLUS::Dist` object.

$bool = $dist->prepare([perl => '/path/to/perl', buildflags => 'EXTRA=FLAGS', force => BOOL, verbose => BOOL])

`prepare` prepares a distribution, running `Module::Build`'s `new_from_context` method, and establishing any prerequisites this distribution has.

When running `Module::Build->new_from_context`, the environment variable `PERL5_CPANPLUS_IS_EXECUTING` will be set to the full path of the `Build.PL` that is being executed. This enables any code inside the `Build.PL` to know that it is being installed via CPANPLUS.

After a succcesfull `prepare` you may call `create` to create the distribution, followed by `install` to actually install it.

Returns true on success and false on failure.

$dist->create([perl => '/path/to/perl', buildflags => 'EXTRA=FLAGS', prereq_target => TARGET, force => BOOL, verbose => BOOL, skiptest => BOOL])

`create` preps a distribution for installation. This means it will run `Build` and `Build test`, via the `Module::Build` API. This will also satisfy any prerequisites the module may have.

If you set `skiptest` to true, it will skip the `Build test` stage. If you set `force` to true, it will go over all the stages of the `Build` process again, ignoring any previously cached results. It will also ignore a bad return value from `Build test` and still allow the operation to return true.

Returns true on success and false on failure.

You may then call `$dist->install` on the object to actually install it.

$dist->install([verbose => BOOL, perl => /path/to/perl])

Actually installs the created dist.

Returns true on success and false on failure.

KNOWN ISSUES

Below are some of the known issues with Module::Build, that we hope the authors will resolve at some point, so we can make full use of Module::Build's power. The number listed is the bug number

on `rt.cpan.org`.

* Module::Build can not be upgraded using its own API (#13169)

This is due to the fact that the Build file insists on adding a path to `@INC` which force the loading of the `not yet installed` Module::Build when it shells out to run it's own build procedure:

* Module::Build does not provide access to install history (#9793)

`Module::Build` runs the create, test and install procedures in it's own processes, but does not provide access to any diagnostic messages of those processes. As an end result, we can not offer these diagnostic messages when, for example, reporting automated build failures to sites like `testers.cpan.org`.

AUTHOR

Originally by Jos Boumans <kane@cpan.org>. Brought to working condition and currently maintained by Ken Williams <kwilliams@cpan.org>.

COPYRIGHT