

NAME

CPANPLUS::Internals::Source

SYNOPSIS

```
### lazy load author/module trees ###

$cb->_author_tree;
$cb->_module_tree;
```

DESCRIPTION

CPANPLUS::Internals::Source controls the updating of source files and the parsing of them into usable module/author trees to be used by CPANPLUS.

Functions exist to check if source files are still `good to use` as well as update them, and then parse them.

The flow looks like this:

```
$cb->_author_tree || $cb->_module_tree
  $cb->_check_trees
    $cb->__check_uptodate
    $cb->_update_source
    $cb->__update_custom_module_sources
    $cb->__update_custom_module_source
  $cb->_build_trees
    $cb->__create_author_tree
    $cb->__retrieve_source
    $cb->__create_module_tree
    $cb->__retrieve_source
    $cb->__create_dslip_tree
    $cb->__retrieve_source
    $cb->__create_custom_module_entries
    $cb->_save_source

$cb->_dslip_defs
```

METHODS

`$cb->_check_trees([update_source => BOOL, path => PATH, verbose => BOOL])`

Retrieve source files and return a boolean indicating whether or not the source files are up to date.

Takes several arguments:

`update_source`

A flag to force re-fetching of the source files, even if they are still up to date.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

`$cb->__check_uptodate(file => $file, name => $name, [update_source => BOOL, verbose => BOOL])`

`__check_uptodate` checks if a given source file is still up-to-date and if not, or when `update_source` is true, will re-fetch the source file.

Takes the following arguments:

`file`

The source file to check.

`name`

The internal shortcut name for the source file (used for config lookups).

`update_source`

Flag to force updating of sourcefiles regardless.

`verbose`

Boolean to indicate whether to be verbose or not.

Returns a boolean value indicating whether the current files are up to date or not.

`$cb->_update_source(name => $name, [path => $path, verbose => BOOL])`

This method does the actual fetching of source files.

It takes the following arguments:

`name`

The internal shortcut name for the source file (used for config lookups).

`path`

The full path where to write the files.

`verbose`

Boolean to indicate whether to be verbose or not.

Returns a boolean to indicate success.

`$cb->_build_trees(uptodate => BOOL, [use_stored => BOOL, path => $path, verbose => BOOL])`

This method rebuilds the author- and module-trees from source.

It takes the following arguments:

`uptodate`

Indicates whether any on disk caches are still ok to use.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

`use_stored`

A boolean flag indicating whether or not it is ok to use previously stored trees. Defaults to true.

Returns a boolean indicating success.

`$cb->__retrieve_source(name => $name, [path => $path, uptodate => BOOL, verbose => BOOL])`

This method retrieves a *storabled* tree identified by `$name`.

It takes the following arguments:

`name`

The internal name for the source file to retrieve.

`uptodate`

A flag indicating whether the file-cache is up-to-date or not.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

Returns a tree on success, false on failure.

`$cb->__save_source([verbose => BOOL, path => $path])`

This method saves all the parsed trees in *storabled* format if *Storable* is available.

It takes the following arguments:

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

Returns true on success, false on failure.

`$cb->__create_author_tree([path => $path, uptodate => BOOL, verbose => BOOL])`

This method opens a source files and parses its contents into a searchable author-tree or restores a file-cached version of a previous parse, if the sources are uptodate and the file-cache exists.

It takes the following arguments:

`uptodate`

A flag indicating whether the file-cache is uptodate or not.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

Returns a tree on success, false on failure.

`$cb->__create_mod_tree([path => $path, uptodate => BOOL, verbose => BOOL])`

This method opens a source files and parses its contents into a searchable module-tree or restores a file-cached version of a previous parse, if the sources are uptodate and the file-cache exists.

It takes the following arguments:

`uptodate`

A flag indicating whether the file-cache is up-to-date or not.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

Returns a tree on success, false on failure.

`$cb->__create_dslip_tree([path => $path, uptodate => BOOL, verbose => BOOL])`

This method opens a source files and parses its contents into a searchable dslip-tree or restores a file-cached version of a previous parse, if the sources are uptodate and the file-cache exists.

It takes the following arguments:

`uptodate`

A flag indicating whether the file-cache is uptodate or not.

`path`

The absolute path to the directory holding the source files.

`verbose`

A boolean flag indicating whether or not to be verbose.

Will get information from the config file by default.

Returns a tree on success, false on failure.

`$cb->_dslip_defs ()`

This function returns the definition structure (ARRAYREF) of the dslip tree.

`$file = $cb->_add_custom_module_source(uri => URI, [verbose => BOOL]);`

Adds a custom source index and updates it based on the provided URI.

Returns the full path to the index file on success or false on failure.

`$index = $cb->__custom_module_source_index_file(uri => $uri);`

Returns the full path to the encoded index file for `$uri`, as used by all `custom module source` routines.

`$file = $cb->_remove_custom_module_source(uri => URI, [verbose => BOOL]);`

Removes a custom index file based on the URI provided.

Returns the full path to the index file on success or false on failure.

`%files = $cb->__list_custom_module_sources`

This method scans the 'custom-sources' directory in your base directory for additional sources to include in your module tree.

Returns a list of key value pairs as follows:

```
/full/path/to/source/file%3Fencoded => http://decoded/mirror/path
```

`$bool = $cb->__update_custom_module_sources([verbose => BOOL]);`

Attempts to update all the index files to your custom module sources.

If the index is missing, and it's a `file://` uri, it will generate a new local index for you.

Return true on success, false on failure.

`$ok = $cb->__update_custom_module_source`

Attempts to update all the index files to your custom module sources.

If the index is missing, and it's a `file://` uri, it will generate a new local index for you.

Return true on success, false on failure.

`$bool = $cb->__write_custom_module_index(path => /path/to/packages, [to => /path/to/index/file, verbose => BOOL])`

Scans the `path` you provided for packages and writes an index with all the available packages to `$path/packages.txt`. If you'd like the index to be written to a different file, provide the `to` argument.

Returns true on success and false on failure.

`$bool = $cb->__create_custom_module_entries([verbose => BOOL])`

Creates entries in the module tree based upon the files as returned by `__list_custom_module_sources`.

Returns true on success, false on failure.