NAME

ExtUtils::Installed - Inventory management of installed modules

SYNOPSIS

```
use ExtUtils::Installed;
my ($inst) = ExtUtils::Installed->new( skip_cwd => 1 );
my (@modules) = $inst->modules();
my (@missing) = $inst->validate("DBI");
my $all_files = $inst->files("DBI");
my $files_below_usr_local = $inst->files("DBI", "all", "/usr/local");
my $all_dirs = $inst->directories("DBI");
my $dirs_below_usr_local = $inst->directory_tree("DBI", "prog");
my $packlist = $inst->packlist("DBI");
```

DESCRIPTION

ExtUtils::Installed provides a standard way to find out what core and module files have been installed. It uses the information stored in .packlist files created during installation to provide this information. In addition it provides facilities to classify the installed files and to extract directory information from the .packlist files.

USAGE

The new() function searches for all the installed .packlists on the system, and stores their contents. The .packlists can be queried with the functions described below. Where it searches by default is determined by the settings found in `%Config::Config`, and what the value is of the PERL5LIB environment variable.

METHODS

Unless specified otherwise all method can be called as class methods, or as object methods. If called as class methods then the "default" object will be used, and if necessary created using the current processes %Config and @INC. See the 'default' option to new() for details.

new()

This takes optional named parameters. Without parameters, this searches for all the installed .packlists on the system using information from `%Config::Config` and the default module search paths `@INC`. The packlists are read using the *ExtUtils::Packlist* module.

If the named parameter `skip_cwd` is true, the current directory `.` will be stripped from `@INC` before searching for .packlists. This keeps ExtUtils::Installed from finding modules installed in other perls that happen to be located below the current directory.

If the named parameter `config_override` is specified, it should be a reference to a hash which contains all information usually found in `%Config::Config`. For example, you can obtain the configuration information for a separate perl installation and pass that in.

```
my $yoda_cfg  = get_fake_config('yoda');
my $yoda_inst =
          ExtUtils::Installed->new(config_override=>$yoda_cfg);
```

Similarly, the parameter `inc_override` may be a reference to an array which is used in place of the default module search paths from `@INC`.

```
use Config;
my @dirs = split(/\Q$Config{path_sep}\E/, $ENV{PERL5LIB});
my $p5libs = ExtUtils::Installed->new(inc_override=>\@dirs);
```

Note: You probably do not want to use these options alone, almost always you will want to set both together.

The parameter `extra_libs` can be used to specify **additional** paths to search for installed modules. For instance

```
my $installed =
         ExtUtils::Installed->new(extra_libs=>["/my/lib/path"]);
```

This should only be necessary if */my/lib/path* is not in PERL5LIB.

Finally there is the 'default', and the related 'default_get' and 'default_set' options. These options control the "default" object which is provided by the class interface to the methods. Setting `default_get` to true tells the constructor to return the default object if it is defined. Setting `default_set` to true tells the constructor to make the default object the constructed object. Setting the `default` option is like setting both to true. This is used primarily internally and probably isn't interesting to any real user.

modules()

This returns a list of the names of all the installed modules. The perl 'core' is given the special name 'Perl'.

files()

This takes one mandatory parameter, the name of a module. It returns a list of all the filenames from the package. To obtain a list of core perl files, use the module name 'Perl'. Additional parameters are allowed. The first is one of the strings "prog", "doc" or "all", to select either just program files, just manual files or all files. The remaining parameters are a list of directories. The filenames returned will be restricted to those under the specified directories.

directories()

This takes one mandatory parameter, the name of a module. It returns a list of all the directories from the package. Additional parameters are allowed. The first is one of the strings "prog", "doc" or "all", to select either just program directories, just manual directories or all directories. The remaining parameters are a list of directories. The directories returned will be restricted to those under the specified directories. This method returns only the leaf directories that contain files from the specified module.

directory_tree()

This is identical in operation to directories(), except that it includes all the intermediate directories back up to the specified directories.

validate()

This takes one mandatory parameter, the name of a module. It checks that all the files listed in the modules .packlist actually exist, and returns a list of any missing files. If an optional second argument which evaluates to true is given any missing files will be removed from the .packlist

packlist()

This returns the ExtUtils::Packlist object for the specified module.

version()

This returns the version number for the specified module.

EXAMPLE

See the example in *ExtUtils::Packlist*.

AUTHOR

Alan Burlison <Alan.Burlison@uk.sun.com>