NAME

Lintian::ProcessableGroup −− A group of objects that Lintian can process

SYNOPSIS

```
 use Lintian::ProcessableGroup;

 my $group = Lintian::ProcessableGroup->new('lintian_2.5.0_i386.changes');
 foreach my $proc ($group->get_processables){
     printf "%s %s (%s)\n", $proc->pkg_name,
            $proc->pkg_version, $proc->pkg_type;
 }
 # etc.
```

DESCRIPTION

Instances of this perl class are sets of processables. It allows at most one source and one changes or buildinfo package per set, but multiple binary packages (provided that the binary is not already in the set).

METHODS

Lintian::ProcessableGroup−>new ([LAB[, FILE]])

Creates a group and optionally add all processables from .changes or .buildinfo file FILE.

If the LAB parameter is given, all processables added to this group will be stored as a lab entry from LAB.

`$group`−>add_new_processable ($pkg_path[, `$pkg_type`])

Adds a new processable of type `$pkg_type` from `$pkg_path`. If `$pkg_type` is not given, it will be determined by the file extension.

This is short hand for:

```
 $group->add_processable(
    Lintian::Processable->new ($pkg_path, $pkg_type));
```

`$group`−>add_processable($proc)

Adds `$proc` to `$group`. At most one source and one changes `$proc` can be in a `$group`. There can be multiple binary `$proc`'s, as long as they are all unique. Successive buildinfo `$proc`'s are silently ignored.

This will error out if an additional source or changes `$proc` is added to the group. Otherwise it will return a truth value if `$proc` was added.

`$group`−>get_processables([$type])

Returns an array of all processables in `$group`. The processables are returned in the following order: changes (if any), source (if any), all binaries (if any) and all udebs (if any).

This order is based on the original order that Lintian processed packages in and some parts of the code relies on this order.

Note if `$type` is given, then only processables of that type is returned.

`$group`−>remove_processable($proc)

Removes `$proc` from `$group`

`$group`−>get_source_processable

Returns the processable identified as the "source" package (e.g. the dsc).

If `$group` does not have a source processable, this method returns `undef`.

`$group`−>get_buildinfo_processable

Returns the processable identified as the "buildinfo" processable (e.g. the buildinfo file).

If `$group` does not have a buildinfo processable, this method returns `undef`.

`$group`−>get_changes_processable

Returns the processable identified as the "changes" processable (e.g. the changes file).

If `$group` does not have a changes processable, this method returns `undef`.

`$group`−>get_binary_processables

Returns all binary (and udeb) processables in `$group`.

If `$group` does not have any binary processables then an empty list is returned.

`$group`−>info

Returns `$info` element for this group.

`$group`−>init_shared_cache

Prepare a shared memory cache for all current members of the group. This is solely a memory saving optimization and is not required for correct performance.

`$group`−>clear_cache

Discard the info element of all members of this group, so the memory used by it can be reclaimed. Mostly useful when checking a lot of packages (e.g. on lintian.d.o).

AUTHOR

Originally written by Niels Thykier <niels@thykier.net> for Lintian.

SEE ALSO

lintian (1)

Lintian::Processable