NAME

MojoMojo::Schema::ResultSet::Page − resultset methods on pages

METHODS

path_pages

```
( $path_pages, $proto_pages ) = __PACKAGE__->path_pages( $path, $id )
```

Accepts a path in URL/Unix directory format, e.g. ‘‘/page1/page2’’. Paths are assumed to be absolute, so a leading slash (/) is not required.

Returns a reference to an array of any pages that exist in the path, starting with ‘‘/’’, and an additional reference to an array of ‘‘proto page’’ hashes for any pages at the end of the path that do not exist. All paths include the root (/), which must exist, so a path of at least one element will always be returned.

The ‘‘proto page’’ hash keys are shown in the example below, where we assume that `/blog` exists and `/blog/My_New_Entry` doesn’t exist yet:

```
{
    depth => 2,
    name => "my_new_entry",
    name_orig => "My_New_Entry",
    path => "/blog/My_New_Entry",
},
```

path_pages_by_id

```
@path_pages = __PACKAGE__->path_pages_by_id( $id )
```

Returns all the pages in the path to a page, given that page’s id.

parse_path

```
@proto_pages = __PACKAGE__->parse_path( $path )
```

Create prototype page objects for each level in a given path.

normalize_name

```
($name_orig, $name) = __PACKAGE__->normalize_name( $name_orig )
```

Strip superfluous spaces, convert the rest to _, then lowercase the result.

resolve_path

```
$an_resolve = __PACKAGE__->resolve_path( %args )
```

Takes the following args:

path_pages
proto_pages
query_pages
current_depth
final_depth

Returns true if the path can be resolved, or false otherwise.

set_paths

```
@pages = __PACKAGE__->set_paths( @pages )
```

Sets the path for multiple pages, either a subtree or a group of non-adjacent pages.

create_path_pages

```
$path_pages = __PACKAGE__->create_path_pages( %args )
```

Find or creates a list of path_pages. Returns a reference to an array of path_pages.

open_gap

```
$parent = __PACKAGE__->open_gap( $parent, $new_page_count )
```

Opens a gap in the nested set numbers to allow the inserting of new pages into the tree. Since nested sets number each node twice, the size of the gap is always twice the number of new pages. Also, since nested sets number the nodes from left to right, we determine what nodes to re-number according to the `rgt` column of the parent of the top-most new node.

Returns a new parent object that is updated with the new `lft` `rgt` nested set numbers.

create_page
Create a new page in the wiki.