NAME

MojoMojo − A Wiki with a tree

SYNOPSIS

```
# Set up database (see mojomojo.conf first)

./script/mojomojo_spawn_db.pl

# Standalone mode

./script/mojomo_server.pl

# In apache conf
<Location /mojomojo>
  SetHandler perl-script
  PerlHandler MojoMojo
</Location>
```

DESCRIPTION

Mojomojo is a content management system, borrowing many concepts from wikis and blogs. It allows you to maintain a full tree-structure of pages, and to interlink them in various ways. It has full version support, so you can always go back to a previous version and see what's changed with an easy diff system. There are also a some of useful features like live AJAX preview while editing, tagging, built-in fulltext search, image galleries, and RSS feeds for every wiki page.

To find out more about how you can use MojoMojo, please visit <http://mojomojo.org/> or read the installation instructions in MojoMojo::Installation to try it out yourself.

METHODS

prepare

Accommodate a forcing of SSL if needed in a reverse proxy setup.

ajax

Return whether the request is an AJAX one (used by the live preview, for example), as opposed to a rgular request (such as one used to view a page).

expand_wikilink

Proxy method for the MojoMojo::Formatter::Wiki expand_wikilink method.

wikiword

Format a wikiword as a link or as a wanted page, as appropriate.

pref

Find or create a preference key. Update it if a value is passed, then return the current setting.

pref_cached

Get preference key/value from cache if possible.

fixw

Clean up wiki words: replace spaces with underscores and remove non−\w, / and . characters.

tz

Convert timezone

prepare_action

Provide ''No DB'' message when one needs to spawn the db (script/mojomojo_spawn.pl).

prepare_path

We override this method to work around some of Catalyst's assumptions about dispatching. Since MojoMojo supports page namespaces (e.g. `/parent_page/child_page`), with page paths that always start with `/`, we strip the trailing slash from `$c->req->base`. Also, since MojoMojo indicates actions by appending a `.$action` to the path (e.g. `/parent_page/child_page.edit`), we remove the page path and save it in `$c->stash->{path}` and reset `$c->req->path` to `$action`. We save the original URI in `$c->stash->{pre_hacked_uri}`.

base_uri
Return `$c->req->base` as an URI object.

uri_for
Override `$c->uri_for` to append path, if a relative path is used.

uri_for_static
`/static/` has been remapped to `/.static/`.

_cleanup_path
Lowercase the path and remove any double-slashes.

_expand_path_elements
Generate all the intermediary paths to `/path/to/a/page`, starting from `/` and ending with the complete path:

```
/
/path
/path/to
/path/to/a
/path/to/a/page
```

get_permissions_data
Permissions are checked prior to most actions, including `view` if that is turned on in the configuration. The permission system works as follows:

1. There is a base set of rules which may be defined in the application config. These are:

   ```
   $c->config->{permissions}{view_allowed} = 1; # or 0
   ```

   Similar entries exist for `delete`, `edit`, `create` and `attachment`. If these config variables are not defined, the default is to allow anyone to do anything.

2. Global rules that apply to everyone may be specified by creating a record with a role id of 0.

3. Rules are defined using a combination of path(s)?, and role and may be applied to subpages or not.

   TODO: clarify.

4. All rules matching a given user's roles and the current path are used to determine the final yes/no on each permission. Rules are evaluated from least-specific path to most specific. This means that when checking permissions on `/foo/bar/baz`, permission rules set for `/foo` will be overridden by rules set on `/foo/bar` when editing `/foo/bar/baz`. When two rules (from different roles) are found for the same path prefix, explicit `allow`s override `deny`s. Null entries for a given permission are always ignored and do not affect the permissions defined at earlier level. This allows you to change certain permissions (such as `create`) only while not affecting previously determined permissions for the other actions. Finally − `apply_to_subpages` `yes`/`no` is exclusive, meaning that a rule for `/foo` with `apply_to_subpages` set to `yes` will apply to `/foo/bar` but not to `/foo` alone. The endpoint in the path is always checked for a rule explicitly for that page − meaning `apply_to_subpages = no`.

user_role_ids
Get the list of role ids for a user.

check_permissions
Check user permissions for a path.

check_view_permission
Check if a user can view a path.

SUPPORT

- <http://mojomojo.org>
- IRC: <irc://irc.perl.org/mojomojo>.
- Mailing list: <http://mojomojo.2358427.n2.nabble.com/>
- Commercial support and customization for MojoMojo is also provided by Nordaaker Ltd. Contact `arneandmarcus AT nordaaker DOT com` for details.

AUTHORS

Marcus Ramberg `marcus AT nordaaker DOT com`

David Naughton `naughton AT umn DOT edu`

Andy Grundman `andy AT hybridized DOT org`

Jonathan Rockway `jrockway AT jrockway DOT us`

A number of other contributors over the years: https://www.ohloh.net/p/mojomojo/contributors

COPYRIGHT

LICENSE