beelance2

jakob notland

Dec 23, 2019

CONTENTS:

class `views.index.`**`Index`**

`GET``()`
Get main page using the projects URL input variable to determine which projects to show.

return index page

class `views.login.`**`Login`**

`GET``()`
Show the login page

return The login page showing other users if logged in

`POST``()`

Log in to the web application and register the session

return The login page showing other users if logged in

`check_rememberme``()`
Validate the rememberme cookie and log in

`login``(`*username*, *userid*, *remember*`)`
Log in to the application

`rememberme``()`
Encode a base64 object consisting of the username signed with the host secret key and the username. Can be reassembled with the hosts secret key to validate user.

return base64 object consisting of signed username and username

classmethod `sign_username``(`*username*`)`

Sign the current users name with the hosts secret key

return The users signed name

class `views.logout.`**`Logout`**

`GET``()`

Log out of the application (kill session and reset variables)

return Redirect to main page

class `views.new_project.`**`New_project`**

`GET``()`
Get the project registration form

return New project page

`POST``()`
Create a new project

return Redirect to main page

`compose_form``(`*data*, *operation*`)`
Compose a new project form by adding or removing a task

param data The data object from web.input

param operation Can be one of the four: add_task, add_user, remove_task, remove user

type operation str

return A project form object with all the required input fields

class `views.open_projects.`**`Open_projects`**

`GET``()`
Get all open projects

return A page containing all open projects

class `views.project.`**`Project`**

`GET``()`
Show info about a single project

return Project info page

class `views.register.`**`Register`**

`GET``()`
Get the registration form

return A page with the registration form

`POST``()`
Handle input data and register new user in database

return Main page

`views.forms.`**`get_apply_form`**`()`

Get the form used to add users to an application and apply

return A form object

`views.forms.`**`get_apply_permissions_form`**`(`*identifier=0, read_permission='TRUE', write_permission='FALSE', modify_permission='FALSE', userid=None*`)`

Get the form used to set permissions for each applicant

param identifier The id of this element

param user_name The current user

param read_permission Permit user to read

param write_permission Permit user to write

param modify_permission Permit user to modify

type identifier int

type user_name str

type read_permission bool

type write_permission bool

type modify_permission bool

return A form object

`views.forms.`**`get_project_form_elements`**(*project_title=''*, *project_description=''*, *category_name=''*)

Generate a set of project form elements

param project_title Project title

param project_description Project description

param category_name Name of the belonging category

type project_title str

type project_description str

type category_name str

return A set of project form elements

`views.forms.`**`get_task_form_elements`**(*identifier=0*, *task_title=''*, *task_description=''*, *budget=''*)

Generate a set of task form elements

param identifier The id of the task

param task_title Task title

param task_description Task description

param budget Task budget

type identifier int, str

type task_title str

type task_description str

type budget int, str

return A set of task form elements

`views.forms.`**`get_user_form_elements`**(*identifier=0*, *user_name=''*, *read_permission=True*, *write_permission=False*, *modify_permission=False*)

Get the user form elements used to set users in project upon creation

param identifier The id of this element

param user_name The current user

param read_permission Permit user to read

param write_permission Permit user to write

param modify_permission Permit user to modify

type identifier int

type user_name str

type read_permission bool

type write_permission bool

type modify_permission bool

return The form elements to add users to a project

`views.utils.`**`get_element_count`**(*data*, *element*)

Determine the number of tasks created by removing the four other elements from count and divide by the number of variables in one task.

param data The data object from web.input

return The number of tasks opened by the client

`views.utils.`**`get_nav_bar`**`(`*session*`)`
Generates the page nav bar

return The navigation bar HTML markup

`models.login.`**`get_user_id_by_name`**`(`*username*`)`

Get the id of the unique username

param username Name of the user

return The id of the user

`models.login.`**`get_user_name_by_id`**`(`*userid*`)`

Get username from user id

param userid The id of the user

return The name of the user

`models.login.`**`get_users`**`()`

Retreive all registrered users from the database

return users

`models.login.`**`match_user`**`(`*username*, *password*`)`
Check if user credentials are correct, return if exists

param username The user attempting to authenticate

param password The corresponding password

type username str

type password str

return user

`models.project.`**`get_categories`**`()`
Get all categories

return List of categories

`models.project.`**`get_project_by_id`**`(`*projectid*`)`
Retrieve a project by its id

param projectid The project id

type projectid str

return The selected project

`models.project.`**`get_projects_by_owner`**`(`*userid*`)`

Retrieve all projects created by a specific user

param userid The id of the user

type userid str

return An array of projects

`models.project.`**`get_projects_by_participant_and_status`**`(`*userid*, *project_status*`)`
Retrieve all projects where the user is a participant with specific status

param userid The id of the participant

param project_status The status to filter on

type userid str

type project_status str

return A list of projects

`models.project.`**`get_projects_by_status_and_category`**(*categoryid*, *project_status*)
Retrieve all projects from a category with a specific status

param catergoryid The id of the category

param project_status The status to filter on

type catergoryid str

type project_status str

return A list of projects

`models.project.`**`get_projects_by_status_and_owner`**(*userid*, *project_status*)
Retrieve all projects owned by a user with a specific status

param userid The id of the owner

param project_status The status to filter on

type userid str

type project_status str

return A list of projects

`models.project.`**`get_task_files`**(*taskid*)
Retrieve all filenames registered in a task :param taskid: The task id :type taskid: str :return: An array of filenames

`models.project.`**`get_tasks_by_project_id`**(*projectid*)
Get all tasks belonging to a project

param project_id The id of the project holding the tasks

type project_id str

return List of tasks

`models.project.`**`get_user_permissions`**(*userid*, *projectid*)

Get permissions for a selected users in a specific project

param userid The id of the user

param projectid The id of the project

type userid str

type projectid str

return Permissions as an array of numbers as boolean values

`models.project.`**`set_project`**(*categoryid*, *userid*, *project_title*, *project_description*, *project_status*)
Store a project in the database

param categoryid The id of the corresponding category

param userid The id of the project owner

param project_title The title of the project

param project_description The project description

param project_status The status of the project

type categoryid str

type userid str

type project_title str

type project_description str

type project_status str

return The id of the new project

`models.project.`**`set_projects_user`**`(`*projectid*, *userid*, *read_permission='TRUE'*, *write_permission='NULL'*, *modify_permission='NULL'*`)`

Add a user to a project with specific permissions

param projectid The project id

param userid The user id

param read_permission Describes whether a user can view information about a project

param write_permission Describes whether a user can add files to tasks

param modify_permission Describes wheter a user can deliver tasks

type projectid str

type userid str

type read_permission str

type write_permission str

`models.project.`**`set_task`**`(`*projectid*, *task_title*, *task_description*, *budget*`)`

Create a task

param projectid The corresponding project id

param task_title The title of the task

param task_description The description of the task

param budget The task budget

type projectid str

type task_title str

type task_description str

type budget str

`models.project.`**`set_task_file`**`(`*taskid*, *filename*`)`

Register a new task - file relationship

param taskid The task id

param filename The name of the file

type taskid str

type filename str

`models.project.`**`update_project_status`**(*projectid*, *status*)

Change the status of a selected project

param projectid The project id

param status The status to change to, should be either open, in progress or finished

type projectid str

type status str

`models.register.`**`set_user`**(*username*, *password*, *full_name*, *company*, *email*, *street_address*, *city*, *state*, *postal_code*, *country*)

Register a new user in the database

param username The users unique user name

param password The password

param full_name The users full name

param company The company the user represents

param email The users email address

param street_address The street address of the user

param city The city where the user lives

param state The state where the user lives

param postal_code The corresponding postal code

param country The users country

type username str

type password str

type full_name str

type company str

type email str

type street_address str

type city str

type state str

type postal_code str

type country str

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX