# accounts package

## Submodules

## accounts.EmailVery module

*class* accounts.EmailVery.**EmailVerity**(*emailAddr*, *username*) [source]

Bases: **threading.Thread**

**run**() [source]

## accounts.admin module

## accounts.models module

*class* accounts.models.**User**(*username*, *password*, *email*, *is_confirmed*) [source]

Bases: **django.db.models.base.Model**

*exception* **DoesNotExist**

Bases: **django.core.exceptions.ObjectDoesNotExist**

*exception* User.**MultipleObjectsReturned**

Bases: **django.core.exceptions.MultipleObjectsReturned**

User.**objects** = <*django.db.models.manager.Manager object*>

User.**usersafety_set**

*class* accounts.models.**UserSafety**(*id*, *user_id*, *activation_key*, *key_expires*) [source]

Bases: **django.db.models.base.Model**

*exception* **DoesNotExist**

Bases: **django.core.exceptions.ObjectDoesNotExist**

*exception* UserSafety.**MultipleObjectsReturned**

Bases: **django.core.exceptions.MultipleObjectsReturned**

UserSafety.**get_next_by_key_expires**(**moreargs*, ***morekwargs*)

UserSafety.**get_previous_by_key_expires**(**moreargs*, ***morekwargs*)

UserSafety.**objects** = <*django.db.models.manager.Manager object*>

UserSafety.**user**

*class* accounts.models.**loginRecord**(*id*, *identity*, *login_time*, *login_ip*, *isSuccess*) [source]

Bases: **django.db.models.base.Model**

*exception* **DoesNotExist**

Bases: **django.core.exceptions.ObjectDoesNotExist**

*exception* loginRecord.**MultipleObjectsReturned**

Bases: **django.core.exceptions.MultipleObjectsReturned**

loginRecord.**get_next_by_login_time**(**moreargs*, ***morekwargs*)

loginRecord.**get_previous_by_login_time**(**moreargs*, ***morekwargs*)

`loginRecord.objects` = *<django.db.models.manager.Manager object>*

# accounts.tests module

# accounts.urls module

# accounts.views module

`accounts.views.createSession`(*request*, *username*, *isAutoLogin*) [source]

create session for the user who has just logged index

Args:

request: the http request which contains session username: the username of the user who has just logged index isAutoLogin: whether the user allow auto login or not

`accounts.views.get_client_ip`(*request*) [source]

`accounts.views.indexView`(*request*) [source]

`accounts.views.isAccountValid`(*username*, *password*) [source]

`accounts.views.isAllowAutoLogin`(*request*) [source]

check is user allow auto login check is user allowed auto login, true or false

Args:

request: the http request to be processed

Returns: A http response which contains json result

`accounts.views.isEmailExists`(*email*) [source]

`accounts.views.isUserExists`(*username*) [source]

`accounts.views.loginAction`(**args*, ***kwargs*) [source]

handle the login request

handle the login request, validate and create session

Args:

request: the http request to be processed

Returns: A http response which contains the json result

`accounts.views.logoutAction`(**args*, ***kwargs*) [source]

`accounts.views.recordLogin`(*username*, *ip*, *isSuccessful*) [source]

`accounts.views.registerAction`(**args*, ***kwargs*) [source]

handle the register request

handle the register request and send confirmation email

Args:

request: the http request

Returns:

return a http response which contents the result json that indicate the register reuslt

`accounts.views.registerSuccessView`(*request*) [source]

`accounts.views.register_confirm`(**args*, ***kwargs*) [source]

finish confirmation and active the account

Args:

request: the http request activation_key: the activation key

Returns:

Http redirect to successful page

`accounts.views.`**`saveNewUser`**(*username*, *password*, *email*) [source]

`accounts.views.`**`testIndexView`**(*request*) [source]

`accounts.views.`**`varifyEmail`**(*email*, *user*, *username*) [source]

email confirmation

makesure the email is valid

Args:

email: the email to be confirmed user: the user of the email

# Module contents