Youniversity Documentation

Release 1.0

The Youniversity Team

December 18, 2015

Contents

CHAPTER 1

Getting started

1. Setup virtual environment:

```
virtualenv venv
source venv/bin/activate
```

2. Setup requirements:

```
pip install -r requirements.txt
```

3. Sync database:

```
python manage.py migrate
```

4. Add user:

```
python manage.py createsuperuser
```

5. Run webserver:

```
python manage.py runserver
```

1.1 Starting the web server

1. Run webserver:

```
python manage.py runserver
```

2. Enter the front page at http://localhost:8000/static/index.html

1.2 Configure Facebook login

1. Enter Django administration page at http://localhost:8000/admin/
2. Click *Sites* then fix the first site name from `example.com` to `localhost:8000`
3. Go back to front page and click *Social applications*
4. Add new social application. Enter the following information:
    - Name: Facebook
    - Client ID, Secret Key: As given by Facebook (Register a new Facebook application)

- Sites: Add `localhost:8000` to the list

CHAPTER 2

API Documentation

The API root is at `/api`

2.1 auth

This endpoint is handled by `authapi.views`

GET /api/auth/check
Check whether the current user is logged in and retrieve information about the user.

This endpoint is handled by `authapi.views.UserViewSet.get()`

Example request:

```
GET /api/auth/check HTTP/1.1
Host: social.whs.in.th
Accept: application/json, text/javascript
Cookie: sessionid=.....
```

Example response when logged in:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS

"whs"
```

Example response when not logged in:

```
HTTP/1.1 403 FORBIDDEN
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS

""
```

Response JSON Object

- **username** (*string*) – Username of current user

Status Codes

- 403 Forbidden – User is not logged in

POST /api/auth/login

Authenticate user in by username/password combination. For Youniversity, it is usually used to authenticate against KU database via IMAP.

This endpoint is handled by `authapi.views.LoginViewSet.post()`

Example request:

```
POST /api/auth/login HTTP/1.1
Host: social.whs.in.th
Accept: application/json, text/plain, */*
Content-Type:application/json;charset=UTF-8

{username: "example", password: "example"}
```

Example of success response:

```
HTTP/1.0 200 OK
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Allow: POST, OPTIONS
Set-Cookie: csrftoken=Q0oxVmaGJUkyIV9tWuaLjl5yySa4HMcE; expires=Sun, 02-Oct-2016 09:37:27 GMT; M
Set-Cookie: sessionid=2cceti4ju0x6t3l8wl62awpdttl6sp2p; expires=Sun, 18-Oct-2015 09:37:27 GMT; h

"whs"
```

Example of failed response:

```
HTTP/1.0 403 FORBIDDEN
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Allow: POST, OPTIONS

{"detail":"Cannot log you in"}
```

Request JSON Object

- **username** (*string*) – Username
- **password** (*string*) – Password

Response JSON Object

- **username** (*string*) – Username of current user

Status Codes

- 403 Forbidden – Cannot log user in with given credentials. The credentials may be invalid, the server may not be able to connect to authentication database, or the user may be disabled.

POST /api/auth/logout

This endpoint is handled by `authapi.views.LogoutViewSet.post()`

Example request:

```
POST /api/auth/logout HTTP/1.1
Host: social.whs.in.th
Accept: application/json, text/javascript
Cookie: sessionid=.....
```

**Example response*:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS
```

2.2 group

This endpoint is handled by `group.views`

GET /api/group/
This endpoint is handled by `group.views.GroupList()`

Example request:

```
GET /api/group HTTP/1.1
Host: social.whs.in.th
Accept: application/json, text/javascript
Cookie: sessionid=.....
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS

[{"name":"test","description":"Description text","short_description":"Requirement to join group"
```

Response JSON Array of Objects

- **name** (*string*) – Group name
- **description** (*string*) – What's this group about
- **short_description** (*string*) – Requirement to join group
- **type** (*int*) – 0 = normal, 1 = classroom

Status Codes

- 401 Unauthorized – Not authenticated

GET /api/group/ (**int:** *id*)
View basic information about a group

This endpoint is handled by `group.views.GroupViewDetail.get()`

Example request:

```
GET /api/group/1 HTTP/1.1
Host: social.whs.in.th
Accept: application/json, text/javascript
Cookie: sessionid=.....
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS
```

```
{"name":"test","description":"Description text","short_description":"Requirement to join group",
```

Response JSON Object

- **name** (*string*) – Group name
- **description** (*string*) – What's this group about
- **short_description** (*string*) – Requirement to join group
- **type** (*int*) – 0 = normal, 1 = classroom
- **member_status** (*int*) – -1 = not a member, 0 = request pending, 1 = member, 2 = administrator

Status Codes

- 404 Not Found – No such group

CHAPTER 3

Modules Index

3.1 User package

3.1.1 Submodules

3.1.2 User.models module

class User.models.**UserProfile**(*id*, *user*, *firstname*, *lastname*, *birthday*, *gender*, *faculty*, *major*, *types*, *country*, *city*, *created*, *picture*, *cover*)

Bases: django.db.models.base.Model

exception DoesNotExist

Bases: django.core.exceptions.ObjectDoesNotExist

exception UserProfile.**MultipleObjectsReturned**

Bases: django.core.exceptions.MultipleObjectsReturned

UserProfile.**get_gender_display**(**moreargs*, ***morekwargs*)

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

UserProfile.**user**

User.models.**create_profile**(*sender*, *instance*, *created*, ***kwargs*)

Create a matching profile whenever a user object is created.

User.models.**user_cover_directory_path**(*instance*, *filename*)

User.models.**user_picture_directory_path**(*instance*, *filename*)

3.1.3 User.serializers module

class User.serializers.**FirstUserProfileSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('user', 'faculty', 'country', 'created')

model

alias of UserProfile

class User.serializers.**FriendShipSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('user', 'created')

model
alias of Friend

class User.serializers.**PictureField**(*read_only=False*, *write_only=False*, *required=None*, *default=<class rest_framework.fields.empty>*, *initial=<class rest_framework.fields.empty>*, *source=None*, *label=None*, *help_text=None*, *style=None*, *error_messages=None*, *validators=None*, *allow_null=False*)

Bases: rest_framework.fields.Field

to_representation(*obj*)

class User.serializers.**UserCoverSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('cover',)

model
alias of UserProfile

class User.serializers.**UserProfilePictureSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('picture',)

model
alias of UserProfile

class User.serializers.**UserProfileSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('user', 'firstname', 'lastname', 'birthday', 'gender', 'faculty', 'major', 'types', 'country', 'city', 'picture',

model
alias of UserProfile

read_only_fields = ('picture', 'cover')

class User.serializers.**UserSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'username', 'first_name', 'last_name', 'profile')

model
alias of *User*

3.1.4 User.views module

3.2 authapi package

This module contains API related to retrieving current user information.

It should be merged into user module later once that is implemented.

3.2.1 Submodules

3.2.2 authapi.views module

3.3 group package

3.3.1 Submodules

3.3.2 group.models module

class group.models.**Group**(*id*, *name*, *gtype*, *type*, *category*, *description*, *short_description*, *activities*, *permisssion*, *date*, *parent*, *cover*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

Group.**category**

Group.**children**

Group.**get_next_by_date**(**moreargs*, ***morekwargs*)

Group.**get_previous_by_date**(**moreargs*, ***morekwargs*)

Group.**groupmember_set**

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

Group.**parent**

class group.models.**GroupCategory**(*id*, *name*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

GroupCategory.**group_set**

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

class group.models.**GroupMember**(*id*, *group*, *user*, *role*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

GroupMember.**create**(*new_group*, *new_user*)

GroupMember.**group**

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

GroupMember.**user**

group.models.**group_cover_directory_path**(*instance*, *filename*)

3.3.3 group.serializers module

class group.serializers.**GroupCategorySerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

app_label = 'group_category'

fields = ('id', 'name')

model
alias of GroupCategory

class group.serializers.**GroupCoverSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

app_label = 'social_group'

fields = ('cover',)

model
alias of Group

class group.serializers.**GroupMemberSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('user', 'role')

model
alias of GroupMember

class group.serializers.**GroupSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

app_label = 'social_group'

extra_kwargs = {'category': {'required': False}}

fields = ('id', 'name', 'description', 'short_description', 'activities', 'type', 'category', 'member_status', 'date', 'm

model
alias of Group

read_only_fields = ('cover',)

class group.serializers.**SubGroupSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

app_label = 'subgroup'

fields = ('id', 'name')

model
alias of Group

class group.serializers.**UserSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'username', 'first_name', 'last_name', 'profile')

model
alias of *User*

3.3.4 group.views module

3.4 newsfeed package

3.4.1 Submodules

3.4.2 newsfeed.models module

class newsfeed.models.**Comment**(*id*, *post*, *user*, *text*, *datetime*, *file*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

Comment.**get_next_by_datetime**(*moreargs, **morekwargs)

Comment.**get_previous_by_datetime**(*moreargs, **morekwargs)

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

Comment.**post**

Comment.**user**

class newsfeed.models.**Post**(*id*, *user*, *text*, *datetime*, *target_type*, *target_id*, *target_name*, *allow_submission*, *pinned*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

Post.**FORMAT**()

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

Post.**comment_set**

Post.**get_next_by_datetime**(*moreargs, **morekwargs)

Post.**get_previous_by_datetime**(*moreargs, **morekwargs)

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

Post.**target_object**
Provides a generic relation to any object through content-type/object-id fields.

Post.**target_type**

Post.**user**

newsfeed.models.**comment_file_name**(*instance*, *filename*)

3.4.3 newsfeed.serializer module

3.4.4 newsfeed.views module

3.5 notification package

3.5.1 Submodules

3.5.2 notification.models module

class notification.models.**Notification**(*id*, *user*, *datetime*, *text*, *target_type*, *target_id*, *link_type*, *link_item*, *reference_detail*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

Notification.**get_next_by_datetime**(**moreargs*, ***morekwargs*)

Notification.**get_previous_by_datetime**(**moreargs*, ***morekwargs*)

Notification.**link_type**

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

Notification.**readed**

Notification.**receiver**

Notification.**target_object**
Provides a generic relation to any object through content-type/object-id fields.

Notification.**target_type**

Notification.**user**

Notification.**usernotification_set**

class notification.models.**UserNotification**(*id*, *notification*, *receiver*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

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

UserNotification.**notification**

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

UserNotification.**receiver**

3.5.3 notification.serializer module

class notification.serializer.**GetNotificationSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'user', 'datetime', 'text', 'target_type', 'target_id', 'link_type', 'link_item', 'reference_detail', 'read

model
alias of Notification

class notification.serializer.**NotificationSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'user', 'datetime', 'text', 'target_type', 'target_id', 'link_type', 'link_item', 'reference_detail', 'recei

model
alias of Notification

class notification.serializer.**TypeSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'model')

model
alias of ContentType

class notification.serializer.**UpdateNotificationSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'user', 'readed')

model
alias of Notification

class notification.serializer.**UserNotificationSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'username', 'first_name', 'last_name')

model
alias of *User*

UserNotificationSerializer.**auto_created = True**

class notification.serializer.**UserProfileSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('picture',)

model
alias of UserProfile

class notification.serializer.**UserSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

class Meta

fields = ('id', 'username', 'first_name', 'last_name', 'profile')

model
alias of *User*

3.5.4 notification.views module

class notification.views.**NotificationView**(***kwargs*)
Bases: rest_framework.views.APIView

This class is an API for get specific notification.

get(*request*)
Get notification of specific user. :param request: Django Rest Framework request object. :param format: pattern for Web APIs.

Returns list of notification of that user.

serializer_class
alias of GetNotificationSerializer

class notification.views.**NotificationViewList**(***kwargs*)
Bases: rest_framework.views.APIView

This class is an API for create and get all notification.

add(*user*, *data*, *receiver_set*, *type*, *link_item*, *reference_detail*, *format=None*)
Create and save notification to database. :param user: user who create notification. :param data: Json information of target of post :param receiver_set: set of receiver who will receive

this notification.

Parameters

- **type** – type of action that create this notification.
- **link_item** – information of action that create this notification.
- **reference_detail** – information of the post target object
- **format** – pattern for Web APIs.

Return:

get(*request*, *format=None*)
Get all of notification in database. :param request: Django Rest Framework request object. :param format: pattern for Web APIs.

Returns list of all notification in database.

serializer_class
alias of NotificationSerializer

class notification.views.**UpdateNotification**(***kwargs*)
Bases: rest_framework.views.APIView

This class is an API for update readed notification.

get(*request*, *noti_id*, *format=None*)
Get notification in database and mark as readed. :param request: Django Rest Framework request object. :param noti_id: id of notification. :param format: pattern for Web APIs.

Returns updated notificaton.

get_object(*noti_id*)
Get notification object by id. :param noti_id: id of notification.

Returns notification object.

serializer_class
alias of `UpdateNotificationSerializer`

update_read(*noti_id*, *format=None*)
Update notification to readed by id. :param noti_if: id of notification. :param format: pattern for Web APIs.

Return:

CHAPTER 4

Indices and tables

- genindex
- modindex
- search
- HTTP Routing Table

/api

```
GET /api/auth/check, 3
GET /api/group/, 5
GET /api/group/(int:id), 5
POST /api/auth/login, 3
POST /api/auth/logout, 4
```

a

g

n

u

A

C

E

F

G

L

M

N