Innopolis Library Installation Guide

Here you can all details about inner working processes of ILMK bot.

scroller.py

File with class for interacting with user, generating 'menu' messages.

class Scroller

1. def **init** (*self*, state, list)

- **Description**: Initializer for object of class.
- **Input**: state - what type of menu is this; list - list with items to scroll through
- **Output**: none

2. def **create_message** (*self*):

- **Description:** creates a message, depending on state, defined during initialization process
- **Input:** none
- **Output:** created message
-

3. def **create_keyboard** (*self*):

- **Description:** creates a keybord(buttons under message) according to a type of request
- **Input:** none
- **Output:** created keyboard

telebot.py

This file contains conversation handlers that help to get user's information and search items.

Handlers - specific functions that trigger actions by commands. Command can be /command or just text. Handlers and commands can also be connected by ConversationHandlers (used with /enroll command and editing medias/values below)

1. def **callback_query_selector** (bot, update)

- **Description:** query handler. Each time user sends inline buttonbot gets a callback query. This handler chooses the appropriate way to run further.
- **Input:** bot - bot object, update - log of bot-user interaction
- **Output:** none

2. def **all_numbers** (input_string)

- **Description:** checks if all the elements of input are digits
- **Input:** input_string - string with information
- **Output:** whether all the elements of input are digits

3. def **ask_name** (bot, update)

- **Description:** handler that starts registration process. Handles from /enroll button. Here user's profile is starting filling
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** next state

4. def **ask_phone** (bot, update)

- **Description:** handler that asks for phone.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** next state

5. def **ask_address** (bot, update)

- **Description:** handler that asks for address.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** next state

6. def **ask_faculty** (bot, update)

- **Description:** handler that asks if the user is faculty member.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** moving to buttons. After buttons moves to end_of_registration.

7. def **end_of_registration** (bot, update)

- **Description:** ends the registraton. If everything is correct, send request to database.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** end of registry_handler

8. def **get_list** (table):

- **Description:** gets the list of all records from the table. Used for update.
- **Input:** table - name of table to update
- **Output:** list with all the records from the table

9. def **create_request_card** (bot, update)

- **Description:** is called when commands are getting called for the first time. Creates a request card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

10. def **create_media_card** (bot, update)

- **Description:** is called when commands are getting called for the first time. Creates a media card.

- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

11. def **create_users_card** (bot, update)

- **Description:** is called when commands are getting called for the first time. Creates a user card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

12. def **issue_media** (bot, update)

- **Description:** is called when commands are getting called for the first time. Creates an issue media card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

13. def **create_log_card** (bot, update)

- **Description:** is called when commands are getting called for the first time. Creates a log card card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

14. def **return_media** (bot, update)

- **Description:** creates return media card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

15. def **edit_users_card** (bot, update)

- **Description:** edits card with users
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

16. def **edit_return_media** (bot, update)

- **Description:** edits return media card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

17. def **edit_request_card** (bot, update)

- **Description:** edits request card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

18. def **edit_media_card** (bot, update)

- **Description:** edits media card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

19. def **edit_issue_media** (bot, update)

- **Description:** edits issue meda card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

20. def **edit_log_card** (bot, update)

- **Description:** edits log card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

21. def **edit_my_medias_card** (bot, update)

- **Description:** updates the previous message with user's media items
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

22. def **librarian_authentication** (user_id)

- **Description:** checks if user is a librarian or not
- **Input:** user_id - Telegram ID of user that we check
- **Output:** whether user is a librarian or not

23. def **librarian_interface** (bot, update)

- **Description: prints a librarian interface**
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

25. def **delete_media** (bot, update, media_id)

- **Description:** deletes media with all its copies
- **Input:** bot - bot object; update - log of bot-user interaction, media_id - ID of media to delete
- **Output:** none

26. def **delete_copy** (bot, update, args)

- **Description:** deletes particular copy of media
- **Input:** bot - bot object; update - log of bot-user interaction, args - id of copy
- **Output:** none

27. def **delete_user** (bot, update, telegram_id)

- **Description:** deletes user
- **Input:** bot - bot object; update - log of bot-user interaction, telegram_id - Telegram ID of user to delete
- **Output:** none

28. def **add_copy** (bot, update, media_id)

- **Description:** adds a copy of particular media item
- **Input:** bot - bot object; update - log of bot-user interaction, media_id - ID of media which copy we add
- **Output:** none

29. def **edit_media** (bot, update, media_id)

- **Description:** edits parameters of media item
- **Input:** bot - bot object; update - log of bot-user interaction, media_id - ID of media which parameters we change
- **Output:** none

30. def **edit_field** (bot, update)

- **Description:** calls the menu to edit media or user's parameter field
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** waits for message with the new value

31. def **change_value** (bot, update)

- **Description:** sends the changes made in **edit_field** to database

- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** end of conversation

32. def **edit_user** (bot, update, telegram_id)

- **Description**: calls a menu with selector (what to edit)
- **Input:** bot - bot object; update - log of bot-user interaction, telegram_id - Telegram ID of user whose parameters needs to be edited
- **Output:** none

33. def **me** (bot, update)

- **Description: prints user's card**
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

34. def **create_new_media** (bot, update)

- **Description:** creates a new media item
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

35. def **create_new_user** (bot, update)

- **Description:** creates a new user
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

36. def **cancel_process** (bot, update)

- **Description: cancels conversations**
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

37. def **confirm_user** (bot, update, args)

- **Description: confirms user with its key**
- **Input:** bot - bot object; update - log of bot-user interaction, arg - key
- **Output:** none

38. register_conversation

- This is a connector which connects command /enroll and many handlers. Each handler asks user for single parameter (phone, address and etc). register_conversation calls handlers one-by-one.

button_actions.py

File that contains all the features which are called when buttons are pressed.

1. def **approve_request** (*self*, bot, update)

- **Description:** approves the request for registration of user
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

2. def **reject_request** (*self*, bot, update)

- **Description:** rejects the request for registration of user
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

3. def **book_media** (*self*, bot, update)

- **Description:** books items
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

4. def **make_return_request** (bot, update, copy_id)

- **Description:** makes a return request
- **Input:** bot - bot object; update - log of bot-user interaction, copy_id - ID of media copy
- **Output:** none

5. def **accept_return** (bot, update, request_id)

- **Description:** accsepts a return request
- **Input:** bot - bot object; update - log of bot-user interaction, request_id - ID of return request
- **Output:** none

6. def **reject_return** (bot, update, request_id)

- **Description:** rejects a return request
- **Input:** bot - bot object; update - log of bot-user interaction, request_id - ID of return request
- **Output:** none

7. def **ask_for_return** (bot, update, copy_id, user_id)

- **Description:** sends a message to user with reqyest of returning a media item
- **Input:** bot - bot object; update - log of bot-user interaction, copy_id - ID of media copy, user_id - ID of user to ask
- **Output:** none

8. def **generate_expiry_date** (media, patron, issue_date)

- **Description:** generates expiry date based on type of media and user
- **Input:** media - media item that is booked; patron - person that books media; issue_date - date of booking
- **Output:** expiry date

9. def **check_copy** (copy_id, user_id)

- **Description:** checks the number of copies of a particular media issued by user
- **Input:** copy_id - ID of media copy, user_id - ID of user
- **Output:** True if number of copies is 0, False otherwise

10. def **convert_to_emoji** (state)

- **Description:** converts states to emojis

- **Input:** state - state to convert
- **Output:** emoji, depending on state

database.py

File that performs connection with database using Pony ORM. It parses record into a class with attribute. Configuration values are hidden in config.py.

key_generator.py

1. def **generate_key** ()

- **Description:** creates a key for user sign in
- **Input:** none
- **Output:** key string