Маратулы Темирболат

Продумать и разработать классы для веб-форума, веб форум должен позволять регистрировать пользователей (имя, логин, пароль), авторизировать пользователя, создавать темы, создавать сообщения внутри темы, выводить список тем, выбирать тему и выводить сообщения в выбранной теме.

import datetime

ONE\_STEP = 1

class User:

    def \_\_init\_\_(self,login,password,user\_name):

        self.\_\_login = login

        self.\_\_password = password

        self.\_\_user\_name = user\_name

    def get\_login(self):

        return self.\_\_login

    def set\_login(self,login):

        self.\_\_login = login

    def get\_password(self):

        return self.\_\_password

    def set\_password(self,password):

        self.\_\_password = password

    def get\_user\_name(self):

        return self.\_\_user\_name

    def set\_user\_name(self,user\_name):

        self.\_\_user\_name = user\_name

class Topic:

    def \_\_init\_\_(self,name,topic\_question,date\_creation,question\_owner):

        self.\_\_name = name

        self.\_\_topic\_question = topic\_question

        self.\_\_date\_creation = date\_creation

        self.\_\_question\_owner = question\_owner

    def get\_name(self):

        return self.\_\_name

    def set\_name(self,name):

        self.\_\_name = name

    def get\_topic\_question(self):

        return self.\_\_topic\_question

    def set\_topic\_question(self,topic\_question):

        self.\_\_topic\_question = topic\_question

    def get\_date\_creation(self):

        return self.\_\_date\_creation

    def set\_date\_creation(self,date\_creation):

        self.\_\_date\_creation = date\_creation

    def get\_question\_owner(self):

        return self.\_\_question\_owner

    def set\_question\_owner(self,question\_owner):

        self.\_\_question\_owner = question\_owner

    def \_\_str\_\_(self):

        return '\nНазвание темы: {}.\n\nВопрос:{}\n\nДата Создания: {}.\

\t  Спросил пользователь: {}\

'.format(self.\_\_name,self.\_\_topic\_question,self.\_\_date\_creation,self.\_\_question\_owner.get\_user\_name())

class Message:

    def \_\_init\_\_(self,text,date\_creation,user\_name):

        self.\_\_text = text

        self.\_\_date\_creation = date\_creation

        self.\_\_user\_name = user\_name

    def get\_text(self):

        return self.\_\_text

    def set\_text(self,text):

        self.\_\_text = text

    def get\_date\_creation(self):

        return self.\_\_date\_creation

    def set\_date\_creation(self,date\_creation):

        self.\_\_date\_creation = date\_creation

    def get\_user\_name(self):

        return self.\_\_user\_name

    def set\_user\_name(self,user\_name):

        self.\_\_user\_name = user\_name

    def \_\_str\_\_(self):

        print('1')

        return '{}.\n\t\t\tОствил пользователь: {} {}\

'.format(self.\_\_text,self.\_\_user\_name,self.\_\_date\_creation)

class View:

    def input\_user\_name(self):

        return input('Введите Имя Пользователя: ')

    def input\_user\_login(self):

        return input('Введите Логин Пользователя: ')

    def input\_date\_creation(self):

        return input('Введите дату: ')

    def input\_user\_password(self):

        return input('Введите Пароль Пользователя: ')

    def input\_topic\_name(self):

        return input('Введите название Темы: ')

    def input\_topic\_message(self):

        return input('Введите сообщение для вашей Темы: ')

    def show\_options(self,options):

        print('-------------------------------------------')

        print()

        current\_option = 1

        for option in options:

            print(current\_option,option)

            current\_option += ONE\_STEP

    def show\_existed\_topics(self,topics):

        print('-------------------------------------------')

        print()

        current\_topic = 1

        for topic in topics:

            print(current\_topic,topic.get\_name())

            current\_topic += ONE\_STEP

    def input\_user\_choice(self):

        return int(input('Ваш выбор: '))

    def show\_non\_existed\_option\_message(self):

        print('\nИзвините, но данной опции не существует')

    def show\_buy\_message(self):

        print('\nСпасибо за использование программы! До свидания!')

    def view\_existed\_user\_message(self):

        print('\nДанный пользователь уже существует')

    def show\_successfully\_created\_user\_message(self):

        print('\nВаш пользователь успешно создан!')

    def view\_occured\_error(self,error):

        print(error)

    def view\_existed\_topic\_message(self):

        print('\nИзвините, но данная тема уже поднята в системе!')

    def view\_non\_existed\_topic\_message(self):

        print('\nИзвините, но такой темы не существует')

    def view\_welcom\_message(self,username):

        print('\nДобро пожаловать в систему',username)

    def view\_successful\_created\_message\_creation(self,created\_item):

        print(created\_item,'успешно создана')

    def view\_created\_item\_structure(self,item):

        print('\n',item)

    def view\_topic\_messages(self,topic\_messages):

        for message in topic\_messages:

            print(message)

class Controller:

    users = [User('Mara\_jc53','12345','temirbolat009kz')]

    topics = []

    topics\_messages = dict()

    current\_user = None

    menu\_options = ['Авторизоваться','Зарегестрироваться','Выйти из программы']

    user\_options = ['Создать новую тему','Создать сообщение внутри темы','Вывести список тем','Вывести список сообщений выбранной темы','Выйти в меню авторизациия']

    view = View()

    def \_\_is\_topic\_exists(self,topic\_name):

        for topic in self.topics:

            if(topic.get\_name() == topic\_name):

                return True

        return False

    def \_\_get\_topic(self,topic\_name):

        for topic in self.topics:

            if(topic.get\_name() == topic\_name):

                return topic

    def create\_new\_topic(self):

        new\_topic = self.view.input\_topic\_name()

        new\_topic\_question = self.view.input\_topic\_message()

        date\_time = datetime.date.today()

        topic\_date\_creation = '{}.{}.{}'.format(date\_time.day,date\_time.month,date\_time.year)

        topic\_owner = self.current\_user

        if(self.\_\_is\_topic\_exists(new\_topic)):

            self.view.view\_existed\_topic\_message()

        else:

            new\_user\_topic = Topic(new\_topic,new\_topic\_question,topic\_date\_creation, topic\_owner)

            self.topics.append(new\_user\_topic)

            self.topics\_messages[new\_user\_topic] = []

            self.view.view\_created\_item\_structure(new\_user\_topic)

            self.view.view\_successful\_created\_message\_creation('\nНовая тема')

    def create\_new\_message\_to\_topic(self):

        topic\_name = self.view.input\_topic\_name()

        if(self.\_\_is\_topic\_exists(topic\_name) == False):

            self.view.view\_non\_existed\_topic\_message()

        else:

            found\_topic = self.\_\_get\_topic(topic\_name)

            message\_text = self.view.input\_topic\_message()

            date\_time = datetime.date.today()

            message\_date\_creation = '{}.{}.{}'.format(date\_time.day,date\_time.month,date\_time.year)

            message\_answered\_user\_name = self.current\_user.get\_user\_name()

            self.topics\_messages[found\_topic].append(Message(message\_text, message\_date\_creation, message\_answered\_user\_name))

            self.view.view\_successful\_created\_message\_creation('Ваше сообщение')

    def view\_topic\_list(self):

        self.view.show\_existed\_topics(self.topics)

    def view\_topic\_messages(self):

        try:

            self.view\_topic\_list()

            user\_choice = self.view.input\_user\_choice()

            if(user\_choice < 1 or user\_choice > len(self.topics)):

                self.view.view\_non\_existed\_topic\_message()

            else:

                selected\_topic = self.topics[user\_choice - ONE\_STEP]

                self.view.view\_created\_item\_structure(selected\_topic)

                self.view.view\_topic\_messages(self.topics\_messages[selected\_topic])

        except Exception as ex:

            self.view.view\_occured\_error(ex)

    enterred\_user\_functions = [create\_new\_topic,create\_new\_message\_to\_topic,view\_topic\_list,view\_topic\_messages]

    def enterred\_user\_menu(self):

        self.view.view\_welcom\_message(self.current\_user.get\_user\_name())

        while True:

            try:

                self.view.show\_options(self.user\_options)

                user\_choice = self.view.input\_user\_choice()

                if(user\_choice < 1 or user\_choice > len(self.user\_options)):

                    self.view.show\_non\_existed\_option\_message()

                    continue

                elif(user\_choice == len(self.user\_options)):

                    break

                self.enterred\_user\_functions[user\_choice - ONE\_STEP](self)

            except Exception as ex:

                self.view.view\_occured\_error(ex)

    def find\_user\_by\_login\_password(self,login,password):

        found\_user = None

        for user in self.users:

            if(user.get\_login() == login and user.get\_password() == password):

                found\_user = user

        return found\_user

    def \_\_is\_user\_exist(self,login):

        for user in self.users:

            if(user.get\_login() == login):

                return True

        return False

    def register\_new\_user(self):

        login = self.view.input\_user\_login()

        password = self.view.input\_user\_password()

        user\_name = self.view.input\_user\_name()

        if(self.\_\_is\_user\_exist(login)):

            self.view.view\_existed\_user\_message()

        else:

            new\_user = User(login,password,user\_name)

            self.users.append(new\_user)

            self.view.show\_successfully\_created\_user\_message()

    def login(self):

        login = self.view.input\_user\_login()

        password = self.view.input\_user\_password()

        found\_user = self.find\_user\_by\_login\_password(login, password)

        if(found\_user is None):

            raise TypeError('Данного пользователя не существует!')

        self.current\_user = found\_user

        self.enterred\_user\_menu()

    registration\_functions = [login,register\_new\_user]

    def main\_menu(self):

        while True:

            try:

                self.view.show\_options(self.menu\_options)

                user\_choice = self.view.input\_user\_choice()

                if (user\_choice < 1 or user\_choice > len(self.menu\_options)):

                    self.view.show\_non\_existed\_option\_message()

                    continue

                elif(user\_choice == len(self.menu\_options)):

                    self.view.show\_buy\_message()

                    break

                self.registration\_functions[user\_choice - ONE\_STEP](self)

            except Exception as ex:

                self.view.view\_occured\_error(ex)

controller = Controller()

controller.main\_menu()

















