Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по домашнему заданию «Тестирование своего Telegram бота»

Выполнил: Ким Алексей Максимович ИУ5-32Б
Дата: 20.12.2021

Москва, 2021 г.

Цель домашнего задания: изучение возможностей создания ботов в Telegram и их тестирования.

Задание:
Модифицируйте код лабораторной работы №6 таким образом, чтобы он был пригоден для модульного тестирования.
Используя материалы лабораторной работы №4 создайте модульные тесты с применением TDD - фреймворка (2 теста) и BDD - фреймворка (2 теста).

Текст программы:
Модифицированный код translate.py:

```python
from aiogram import Bot, Dispatcher, executor, types
from aiogram.dispatcher.filters import state

from bot import dp
from pasring import parsing
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_currency_names = ['USD', 'EUR', 'RUB']
data = []
a = parsing()

class chosen_curr(StatesGroup):
    waiting_for_translate_curr1 = State()
    waiting_for_translate_curr2 = State()
    waiting_for_translate_num = State()


async def transtale_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_currency_names:
        keyboard.add(name)
    await message.answer("Выберите валюту с какой хотите поменять: ", reply_markup=keyboard)
    await chosen_curr.waiting_for_translate_curr1.set()


async def translate_curr1_chosen(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[0] = message.text
    await chosen_curr.next()
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for name in available_currency_names:
        keyboard.add(name)
    await message.answer("Выберите вторую валюту: ")
    await chosen_curr.waiting_for_translate_curr2.set()

async def translate_curr2_chosen(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[1] = message.text
    await chosen_curr.next()
    await message.answer("Напишите сумму, которое нужно перевести: ", reply_markup=types.ReplyKeyboardRemove())
```

```
    await chosen_curr.waiting_for_translate_num.set()

def raschet(a, b, c):
    return a * (b/c)
async def translate(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data[2] = float(message.text)
        await chosen_curr.next()
    async with state.proxy() as data:
        if data[0] == "USD":
            d = float(a[1].replace(',', '.'))
            if data[1] == "EUR":
                e = float(a[2].replace(',', '.'))
            else:
                e = 1
            f = raschet(data[2], d, e)
        elif data[0] == "EUR":
            d = float(a[2].replace(',', '.'))
            if data[1] == "USD":
                e = float(a[1].replace(',', '.'))
            else:
                e = 1
            f = data[2] * (d / e)
        elif data[0] == "RUB" and data[1] == "USD":
            d = float(a[1].replace(',', '.'))
            f = data[2]/d
        elif data[0] == "RUB" and data[1] == "EUR":
            d = float(a[2].replace(',', '.'))
            f = data[2] / d
    await message.answer(f'Переводим {data[2]} {data[0]} в {data[1]} и получаем {f} {data[1]}')
    await state.finish()


def register_handlers_translate(dp: Dispatcher):
    dp.register_message_handler(transtale_start, commands=['translate'], state=None)
    dp.register_message_handler(translate_curr1_chosen, state=chosen_curr.waiting_for_translate_curr1)
    dp.register_message_handler(translate_curr2_chosen, state=chosen_curr.waiting_for_translate_curr2)
    dp.register_message_handler(translate, state=chosen_curr.waiting_for_translate_num)
```

Test_bot.py:

```
import unittest
from translate import raschet
class TestRoots(unittest.TestCase):

    def test_roots_is_equal(self):
        self.assertEqual(raschet(73.35, 83.16, 100), 88.2034632034632)

    def test_string_root(self):
        self.assertRaises(TypeError, raschet, "pepa")

if __name__ == '__main__':
    unittest.main()
```

Вывод программы:

```
Ran 1 test in 0.002s

OK

Process finished with exit code 0
```