API: Telegram-боты | Я.Шпора

Бот — программа, взаимодействующая с пользователем через **мессенджер**, систему обмена сообщениями. Большинство современных мессенджеров предоставляют своим пользователям API.

API в Telegram

У мессенджера Telegram есть два API — **Client API** и **Bot API.**

- **Client API** — это программный интерфейс для управления аккаунтом Telegram: через этот интерфейс можно от имени аккаунта отправлять сообщения, вступать в группы или изменять информацию в профиле.
- **Bot API** предназначен для управления с ботами, созданными через Telegram-бота `@BotFather`.

Client API

Для создания программных клиентов в Python есть несколько популярных библиотек, например, **pyrogram**.

Установка pyrogram:

```
pip install pyrogram==2.0.106
```

Чтобы получить доступ к Client API нужно создать (фактически — зарегистрировать) своё приложение на странице https://my.telegram.org/. После регистрации можно будет получить ключи `api_id` и `api_hash` — идентификаторы вашего приложения. Для одного Телеграм-аккаунта можно создать только одно приложение.

Пример программного клиента, написанного с помощью библиотеки pyrogram:

```
# /client_api/main.py

from pyrogram import Client

api_id = <Ваш api_id>
api_hash = '<Ваш api_hash>'

# Создаём программный клиент, передаём в него
```

```
# имя сессии и данные для аутентификации в Client API
app = Client('my_account', api_id, api_hash)

app.start()
# Отправляем сообщение
# Первый параметр - это id чата или имя получателя.
# Зарезервированное слово 'me' означает собственный аккаунт отпра
app.send_message('me', 'Привет, это я!')
app.stop()
```

При первом запуске программы в командной строке будет запрошен номер телефона, на который зарегистрирован ваш Telegram-аккаунт. Введите номер в консоль, после этого на телефон или в Telegram придёт проверочный код.

Bot API

Bot API — это API, предназначенный для управления ботами в мессенджере Telegram. Программа, управляющая ботом, взаимодействует с **Bot API** через HTTP-запросы: их можно отправлять и с помощью Python-библиотеки *requests*, и посредством специальных библиотек-обёрток.

Простые запросы можно делать и вручную, через обычный браузер. При каждом запросе к **Bot API** должен передаваться токен, который выдаётся при создании бота.

Регистрация и настройка бота

Начните диалог с ботом *@BotFather,* отправьте команду `/newbot` и укажите параметры нового бота — имя, видимое пользователям и техническое имя. Технические имена ботов должны быть уникальны.

Настройка аккаунта бота: в *@BotFather* отправьте команду `/mybots`; в ответ вы получите список ботов, которыми вы управляете. Укажите бота, которого хотите отредактировать, и нажмите кнопку **Edit Bot**.

Библиотека pyTelegramBotAPI для Bot API

В **pyTelegramBotAPI** все методы **Bot API** вызываются как методы различных классов библиотеки.

Установка библиотеки:

```
pip install pyTelegramBotAPI==4.14.1
```

Отправка сообщений: класс Bot()

Класс **TeleBot()** реализует все методы API, связанные с отправкой, редактированием, пересылкой или удалением сообщений и с прочими активными действиями бота.

Пример кода для отправки сообщения:

```
from telebot import TeleBot

# Укажите токен,
# который вы получили от @Botfather при создании бот-аккаунта:
bot = TeleBot(token='<token>')
# Укажите id своего аккаунта в Telegram:
chat_id = <chat_id>
text = 'Вам телеграмма!'
# Вызываем метод send_message, с помощью этого метода отправляются
bot.send_message(chat_id, text)
```

Обработка входящих сообщений

Сообщения в Telegram делятся на несколько типов:

- простые текстовые сообщения,
- сообщения-картинки,
- стикеры,
- файлы,
- аудио

…и прочие.

Для обработки каждого типа сообщений используется отдельный **хендлер** (от англ. *handler*, «обработчик»).

Чтобы преобразовать обычную функцию в хендлер, в **pyTelegramBotApi** применяется декоратор `@bot . message_handler`, где `bot` — это объект бота, экземпляр класса `TeleBot`.

В коде программы-бота может быть несколько хендлеров. Каждое полученное сообщение анализируется — и вызывается хендлер, наиболее подходящий для конкретного сообщения.

Порядок создания хендлера:

1. Объявить функцию-обработчик.
2. Обернуть функцию-обработчик декоратором `@bot.message_handler`.

Декоратор `@bot.message_handler` передаёт в декорируемую функцию аргумент `message` — объект, в котором хранятся все данные о полученном сообщении: текст сообщения, id отправителя и много другой полезной информации. Функция обработчик должна ожидать этот аргумент.

```
from telebot import TeleBot  # Импортируем пакет.


bot = TeleBot(token='<token>')  # Создаём объект бота с токеном.

@bot.message_handler()  # Задекорировали.
def say_hi(message):    # Функция say_hi() ожидает на вход объект
    ...
```

Получение сообщений

Существует два способа получения сообщений ботом — polling и webhooks.

Polling: программа регулярно отправляет запросы к серверу, чтобы проверить, нет ли новых данных.

Webhooks: методика «обратных вызовов»: сервер Telegram сам уведомляет бота о появившихся обновлениях.

Запуск механизма поллинга с помощью `bot.polling()`:

```
from telebot import TeleBot


bot = TeleBot(token='<token>')

@bot.message_handler(content_types=['text'])
```

```
def say_hi(message):
    chat = message.chat
    chat_id = chat.id
    bot.send_message(chat_id=chat_id, text='Привет, я KittyBot!')


# Метод polling() запускает процесс; приложение начнёт
# отправлять регулярные запросы для получения входящих сообщений.
bot.polling()
```

При обработке входящего сообщения важен порядок обработчиков: узкоспециализированные обработчики в коде должны располагаться выше, чем обработчики «общего назначения».

Объект message

Чтобы правильно среагировать на сообщение пользователя и отправить ему ответ — бот должен знать `chat_id` этого пользователя. Получить идентификатор чата можно из объекта `message`.

```
chat = message.chat
chat_id = chat.id  # Получение идентификатора чата.
```

`message` — объект, который содержит полную информацию о чате с пользователем:

```
{
    'content_type': 'text',
    'id': 78,
    'message_id': 78,
    'from_user': {
        'id': 51652xxx,
        'is_bot': False,
        'first_name': 'Stas',
        'username': 'stasyan',
        'last_name': 'Basov',
        'language_code': 'ru',
        'can_join_groups': None,
        'can_read_all_group_messages': None,
```

```
        'supports_inline_queries': None,
        'is_premium': None,
        'added_to_attachment_menu': None,
    },
    'text': '/start',
    'date': 1705563111,
    'chat': {
        'id': 51652xxx,
        'type': 'private',
        'title': None,
        'username': 'stasyan',
        'first_name': 'Stas',
        ...
    },
    ...
}
```

В итоге автоответчик на текстовые сообщения может выглядеть так:

```
from telebot import TeleBot


bot = TeleBot(token='<token>')

@bot.message_handler(commands=['start'])
def wake_up(message):
    print(message)  # Выводим на печать объект message
    chat = message.chat
    chat_id = chat.id
    bot.send_message(chat_id=chat_id, text='Спасибо, что включили


@bot.message_handler(content_types=['text'])
def say_hi(message):
    chat = message.chat
    chat_id = chat.id
    bot.send_message(chat_id=chat_id, text='Привет, я KittyBot!')
```

```
bot.polling()
```

Обработка команд

Декоратор `@bot.message_handler` можно настроить на отслеживание сообщений с **командами** — так называют сообщения, начинающихся со слеша `/`. Команды могут состоять из латинских букв, цифр и символа подчёркивания; названия командам даёт разработчик. Самые распространённые команды — `/start`, `/help`.

Пример расширения команды `/start`.

```
from telebot import TeleBot


bot = TeleBot(token='<token>')

# В параметры декоратора передаём список команд,
# которые будет обрабатывать хендлер. Слеш перед именем команды не
@bot.message_handler(commands=['start'])
def wake_up(message):
    print(message)  # Выводим на печать объект message
    chat = message.chat
    chat_id = chat.id
    bot.send_message(chat_id=chat_id, text='Спасибо, что включили


...

bot.polling()
```

Кнопки вместо текстового ввода

Кнопки в интерфейс бота добавляются с помощью класса `types.ReplyKeyboardMarkup`. С помощью метода `.add()` объекта этого класса можно добавить сколько угодно кнопок на клавиатуру.

Порядок действий:

- создать объект клавиатуры;
- создать объекты кнопок;
- добавить кнопки на клавиатуру.

При добавлении кнопок можно расположить их в несколько рядов, явно указав их расположение или просто ограничив количество кнопок в ряду.

Пример добавления нескольких кнопок в интерфейс бота:

```
...
from telebot import TeleBot, types
...

@bot.message_handler(commands=['start'])
def wake_up(message):
    chat = message.chat
    name = message.chat.first_name
    # Создаем объект клавиатуры:
    keyboard = types.ReplyKeyboardMarkup()
    # Создаем объекты кнопок:
    button_newcat = types.KeyboardButton('/newcat')
    button_start = types.KeyboardButton('/start')
    # Добавляем объекты кнопок на клавиатуру:
    keyboard.add(button_newcat)
    keyboard.add(button_start)

    bot.send_message(
        chat_id=chat.id,
        text=f'Привет, {name}. Смотри, какие команды доступны',
        # Отправляем клавиатуру в сообщении бота: передаём объект
        # в параметр reply_markup объекта send_message.
        # Telegram-клиент "запомнит" клавиатуру и будет отображать
        reply_markup=keyboard,
    )

...
```

Отправка изображений от имени бота

Для отправки изображения используется метод `.send_image()` класса `Bot`.

```
from telebot import TeleBot

bot = TeleBot(token='<token>')

URL = 'https://cdn2.thecatapi.com/images/3dl.jpg'  # Ссылка на изо

chat_id = <chat_id>
text = 'Вам телеграмма!'
# Отправка сообщения:
bot.send_message(chat_id, text)
# Отправка изображения:
bot.send_photo(chat_id, URL)
```

Получатель увидит в клиенте не ссылку, а изображение.