Материалы к занятию

Bot API представляет из себя HTTP-интерфейс для работы с ботами в Telegram.
Каждому боту при создании присваивается уникальный токен вида *123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew71*.

Все запросы к Telegram Bot API должны осуществляться через HTTPS в следующем виде: https://api.telegram.org/bot<token>/НАЗВАНИЕ_МЕТОДА. Например: *https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe*

Допускаются GET и POST запросы. Для передачи параметров в Bot API доступны 4 способа:

- Запрос в URL
- application/x-www-form-urlencoded
- application/json (не подходит для загрузки файлов)
- multipart/form-data (для загрузки файлов)
-

Ответ придёт в виде JSON-объекта, в котором всегда будет булево поле ok и опциональное строковое поле description, содержащее человекочитаемое описание результата. Если поле ok истинно, то запрос прошёл успешно и результат его выполнения можно увидеть в поле result. В случае ошибки поле ok будет равно false, а причины ошибки будут описаны в поле description.

Существует два диаметрально противоположных по логике способа получать обновления от вашего бота: getUpdates и вебхуки. Входящие обновления будут храниться на сервере до тех пор, пока вы их не обработаете, но не дольше 24 часов.

Независимо от способа получения обновлений, в ответ вы получите объект Update, преобразованный в JSON.

Update - объект представляет из себя входящее обновление. Под обновлением подразумевается действие, совершённое с ботом — например, получение сообщения от пользователя.
Только один из необязательных параметров может присутствовать в каждом обновлении.

Рассмотрим параметры данного объекта:

update_id - Уникальный идентификатор обновления. Идентификаторы обновления начинаются с определенного положительного числа и последовательно увеличиваются. Этот идентификатор становится особенно удобным, если используются Webhooks, поскольку он позволяет вам игнорировать повторяющиеся обновления или восстанавливать правильную последовательность обновлений, если происходит сбой.

message - Новое входящее сообщение любого типа — текст, фотография, стикер и т.д. (Необязательный параметр)

inline_query - Новый входящий встроенный запрос.(Необязательный параметр)

chosen_inline_result - Результат встроенного запроса, который был выбран пользователем и отправлен его собеседнику.(Необязательный параметр)

callback_query - Новый входящий встроенный запрос.(Необязательный параметр)

getUpdates - Этот метод используется для получения обновлений через long polling. Ответ возвращается в виде массива объектов Update. Long Polling — это технология, которая позволяет получать данные о новых событиях с помощью «длинных запросов». Сервер получает запрос, но отправляет ответ на него не сразу, а лишь тогда, когда произойдет какое-либо событие (например, придёт новое сообщение), либо истечет заданное время ожидания.

offset - Идентификатор первого обновления, которое должно быть возвращено. Должно быть больше на единицу, чем самый высокий из идентификаторов ранее полученных обновлений. По умолчанию возвращаются обновления, начинающиеся с самого раннего неподтвержденного обновления. Обновление считается подтвержденным, как только вызывается getUpdates со смещением, превышающим его update_id. Отрицательное смещение может быть задано для получения обновлений, начиная с -offset update, из конца очереди обновлений. Все предыдущие обновления будут забыты.

limit - Ограничивает количество извлекаемых обновлений. Принимаются значения в диапазоне от 1 до 100. Значение по умолчанию равно 100.

timeout - Тайм-аут в секундах для длительного опроса. Значение по умолчанию равно 0, т.е. обычный короткий опрос

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'


def get_updates():
   r = requests.get(f'{BASE_URL}{TOKEN}/getUpdates')
   message = r.json()['result'][-1]['message']['text']
```

```
import requests
```

```
BASE_URL = 'https://api.telegram.org/bot'

TOKEN = '5580946066:AAH9SSwpAd_Pyp9pNkE9NrlD7DDZ53igeio'


def get_updates():
   r = requests.get(f'{BASE_URL}{TOKEN}/getUpdates?offset=1&timeout=10')
   print(r.json())


get_updates()
```

1. Этот метод не будет работать, если у вас уже подключен webhook.
2. Во избежания повторяющихся обновлений, рекомендуется высчитывать offset каждый раз заново.

setWebhook - метод необходим для задания URL вебхука, на который бот будет отправлять обновления. Каждый раз при получении обновления на этот адрес будет отправлен HTTPS POST с сериализованным в JSON объектом Update. При неудачном запросе к вашему серверу попытка будет повторена умеренное число раз.

Для большей безопасности рекомендуется включить токен в URL вебхука, например, так: *https://yourwebhookserver.com/<token>*.
Так как никто посторонний не знает вашего токена, вы можете быть уверены, что запросы к вашему вебхуку шлёт именно Telegram.
Вебхук – это механизм оповещения о происходящих в системе событиях посредством функций обратных вызовов. Когда случается интересующее клиента событие, сервер отправляет HTTP-запрос на URL-адрес, предоставленный клиентом для приема вебхуков.

url - HTTPS url для отправки запросов. Чтобы удалить вебхук, отправьте пустую строку.
certificate - Загрузка публичного ключа для проверки корневого сертификата.

1. При подключенном и настроенном вебхуке метод getUpdates не будет работать.
2. При использовании самоподписанного сертификата, вам необходимо загрузить публичный ключ с помощью параметра certificate.
3. На текущий момент отправка обновлений через вебхуки доступна только на эти порты: 443, 80, 88, 8443.

getWebhookInfo - Содержит информацию о текущем состоянии вебхука

url - URL вебхука, может быть пустым

has_custom_certificate - True, если вебхук использует самозаверенный сертификат

pending_update_count - Количество обновлений, ожидающих доставки

С вебхуками подробнее мы познакомимся позже.

Рассмотрим доступные типы согласно официальной документации.Все типы, использующиеся в Bot API, являются JSON-объектами.

User - этот объект представляет бота или пользователя Telegram.

id - Уникальный идентификатор пользователя или бота

first_name - Имя пользователя или бота

last_name - Фамилия бота или пользователя (Необязательный параметр)

username - username бота или пользователя (Необязательный параметр)

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'


def get_updates():
   r = requests.get(f'{BASE_URL}{TOKEN}/getUpdates')
   print(r.json()['result'][-1]['message']['from'])


get_updates()
```

Результат
{'id': 1800000000,'is_bot': False,'first_name': 'Иван,'username': 'ivanivanov','language_code': 'ru'}

Chat - этот объект представляет собой чат.

id - Уникальный id чата

type - Тип чата: “private”, “group”, “supergroup” или “channel”

title - Название, для каналов или групп (Необязательный параметр)

username - Username, для чатов и некоторых каналов (Необязательный параметр)

first_name - Имя собеседника в чате(Необязательный параметр)

last_name - Фамилия собеседника в чате(Необязательный параметр)

all_members_are_administrators - True, если все участники чата являются администраторами(Необязательный параметр)

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'


def get_updates():
   r = requests.get(f'{BASE_URL}{TOKEN}/getUpdates')
   print(r.json()['result'][-1]['message']['chat'])


get_updates()
```

Результат
{'id': 1800000000,'first_name': 'Иван','username': 'ivanivanov','type': 'private'}

Message - этот объект представляет собой сообщение

message_id - Уникальный идентификатор сообщения
from - Отправитель. Может быть пустым в каналах.(Необязательный параметр)
date - Дата отправки сообщения (Unix time)
chat - Диалог, в котором было отправлено сообщение.
forward_from - Для пересланных сообщений: отправитель оригинального сообщения.(Необязательный параметр)
forward_date - Для пересланных сообщений: дата отправки оригинального сообщения(Необязательный параметр)
reply_to_message - Для ответов: оригинальное сообщение(Необязательный параметр)
text - Для текстовых сообщений: текст сообщения, 0-4096 символов(Необязательный параметр)
entities - Для текстовых сообщений: особые сущности в тексте сообщения.(Необязательный параметр)
audio - Информация об аудиофайле(Необязательный параметр)
document - Информация о файле(Необязательный параметр)
photo - Доступные размеры фото(Необязательный параметр)

sticker - Информация о стикере(Необязательный параметр)
video - Информация о видеозаписи(Необязательный параметр)
voice - Информация о голосовом сообщении(Необязательный параметр)
caption - Подпись к файлу, фото или видео, 0-200 символов(Необязательный параметр)
contact - Информация об отправленном контакте(Необязательный параметр)
location - Информация о местоположении(Необязательный параметр)
venue - Информация о месте на карте(Необязательный параметр)
new_chat_member - Информация о пользователе, добавленном в группу(Необязательный параметр)
left_chat_member - Информация о пользователе, удалённом из группы(Необязательный параметр)
new_chat_title - Название группы было изменено на это поле(Необязательный параметр)
new_chat_photo - Фото группы было изменено на это поле(Необязательный параметр)
delete_chat_photo - Сервисное сообщение: фото группы было удалено(Необязательный параметр)
group_chat_created - Сервисное сообщение: группа создана(Необязательный параметр)
supergroup_chat_created - Сервисное сообщение: супергруппа создана(Необязательный параметр)
channel_chat_created - Сервисное сообщение: канал создан(Необязательный параметр)
migrate_to_chat_id - Группа была преобразована в супергруппу с указанным идентификатором(Необязательный параметр)
migrate_from_chat_id - Супергруппа была создана из группы с указанным идентификатором(Необязательный параметр)
pinned_message - Указанное сообщение было прикреплено(Необязательный параметр)

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = '5580946066:AAH9SSwpAd_Pyp9pNkE9NrlD7DDZ53igeio'


def get_updates():
   r = requests.get(f'{BASE_URL}{TOKEN}/getUpdates')
   print(r.json()['result'][-1]['message'])


get_updates()
```

Результат

{'message_id': 102, 'from': {'id': 1800000000, 'is_bot': False, 'first_name': 'Иван', 'username': 'ivanivanov', 'language_code': 'ru'}, 'chat': {'id': 1800000000, 'first_name': 'Иван', 'username': 'ivanivanov', 'type': 'private'}, 'date': 1661169078, 'text': 'hhhhh'}

Давайте напишем прототип бота, который получает сообщения, проверяет новое ли оно и если новое, а также если это написал пользователь-администратор и текст равен /start, он будет отсылать нам обратно сообщение с Привет username

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'
ADMINS = [Ваш id, ]
def pulling():
   count_message = 0
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           user_id = message['message']['from']['id']
           user_name = message['message']['from']['username']
           text = message['message']['text']
           if user_id in ADMINS and text == '/start':

requests.get(f'{BASE_URL}{TOKEN}/sendMessage?chat_id={user_id}&text=Привет
{user_name}')

pulling()
```

Учитывая, что получаем мы словарь и зная, как работать со словарями, мы сохраним необходимые значения в переменные. Так мы получили username, id пользователя и текст, который он прислал

PhotoSize - Этот объект представляет изображение определённого размера или превью файла / стикера.

file_id - Уникальный идентификатор файла
width - ширина фото
height - высота фото
file_size - размер файла

Переделаем наш код. Предположим администратор прислал нам изображение. В ответ мы отошлем его ему обратно. Для этого воспользуемся методом sendPhoto

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'
ADMINS = [Ваш id, ]
def pulling():
   count_message = 0
```

```
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           file_id = message['message']['photo'][-1]['file_id']
           caption = message['message']['caption']
           user_id = message['message']['from']['id']
           user_name = message['message']['from']['username']
           if user_id in ADMINS:

requests.get(f'{BASE_URL}{TOKEN}/sendPhoto?chat_id={user_id}&photo={file_id
}&caption={caption}')

pulling()
```

sendPhoto - Отправляет фото указанному пользователю

chat_id - Уникальный идентификатор целевого чата или имя пользователя целевого канала
photo - Фото для отправки. Вы можете либо передать file_id в виде строки, чтобы повторно отправить фотографию, которая уже есть на серверах Telegram, либо загрузить новую фотографию, используя multipart/form-data.
caption - описание фото
disable_notification - Отправляет сообщение беззвучно. Пользователи iOS не получат уведомление, пользователи Android получат уведомление без звука.
reply_to_message_id - Если сообщение является ответом, идентификатор исходного сообщения
reply_markup - Дополнительные опции интерфейса. Объект, сериализованный в формате JSON для встроенной клавиатуры, пользовательской клавиатуры ответа, инструкций по скрытию клавиатуры ответа или принудительному получению ответа от пользователя.

Audio - Этот объект представляет аудиозапись, которую клиенты Telegram воспринимают как музыкальный трек.

file_id - Уникальный идентификатор файла
duration - Длительность в секундах
performer - Исполнитель аудио, как определено отправителем или звуковыми тегами
title - Название аудио
mime_type - MIME файла, заданный отправителем
file_size - Размер файла

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'
ADMINS = [Ваш id, ]
def pulling():
   count_message = 0
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           file_id = message['message']['audio']['file_id']
           user_id = message['message']['from']['id']
           if user_id in ADMINS:

requests.get(f'{BASE_URL}{TOKEN}/sendAudio?chat_id={user_id}&audio={file_id
}')

pulling()
```

Document - Этот объект представляет файл, не являющийся фотографией, голосовым сообщением или аудиозаписью.

file_id - уникальный id

thumb - Эскиз документа, определенный отправителем

file_name - Исходное имя файла, определенное отправителем

mime_type - MIME файла, заданный отправителем

file_size - Размер файла

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = '5580946066:AAH9SSwpAd_Pyp9pNkE9NrlD7DDZ53igeio'
ADMINS = [1865314469, ]
def pulling():
   count_message = 0
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           file_id = message['message']['document']['thumb']['file_id']
           user_id = message['message']['from']['id']
           if user_id in ADMINS:

requests.get(f'{BASE_URL}{TOKEN}/sendDocument?chat_id={user_id}&document={f
ile_id}')
```

```
pulling()
```

sendDocument - Используйте этот метод для отправки общих файлов. При успешном выполнении возвращается отправленное сообщение. В настоящее время боты могут отправлять файлы любого типа размером до 50 МБ, это ограничение может быть изменено в будущем.

chat_id - Уникальный id
document - Файл для отправки. Вы можете либо передать file_id в виде строки, чтобы повторно отправить файл, который уже находится на серверах Telegram, либо загрузить новый файл, используя multipart/form-data.
caption - Заголовок документа (также может использоваться при повторной отправке документов по file_id), 0-200 символов
disable_notification - Отправляет сообщение беззвучно. Пользователи iOS не получат уведомление, пользователи Android получат уведомление без звука.
reply_to_message_id - Если сообщение является ответом, идентификатор исходного сообщения
reply_markup - Дополнительные опции интерфейса. Объект, сериализованный в формате JSON для встроенной клавиатуры, пользовательской клавиатуры ответа, инструкций по скрытию клавиатуры ответа или принудительному получению ответа от пользователя.

Sticker - этот объект представляет стикер.

file_id - Уникальный идентификатор файла
width - Ширина стикера
height - Высота стикера
thumb - Превью стикера в формате .webp или .jpg
file_size - Размер файла

sendSticker - этот метод для отправки стикеров

chat_id - Уникальный идентификатор
sticker - Наклейка для отправки. Вы можете либо передать file_id в виде строки для повторной отправки стикера, который уже есть на серверах Telegram, либо загрузить новый стикер, используя multipart/form-data.
disable_notification - Отправляет сообщение беззвучно. Пользователи iOS не получат уведомление, пользователи Android получат уведомление без звука.
reply_to_message_id - Если сообщение является ответом, идентификатор исходного сообщения

reply_markup - Дополнительные опции интерфейса. Объект, сериализованный в формате JSON для встроенной клавиатуры, пользовательской клавиатуры ответа, инструкций по скрытию клавиатуры ответа или принудительному получению ответа от пользователя.

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = 'Ваш токен'
ADMINS = [Ваш id, ]
def pulling():
   count_message = 0
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           file_id = message['message']['sticker']['file_id']
           user_id = message['message']['from']['id']
           if user_id in ADMINS:

requests.get(f'{BASE_URL}{TOKEN}/sendSticker?chat_id={user_id}&sticker={fil
e_id}')

pulling()
```

Video – этот объект представляет видеозапись.

file_id - Уникальный идентификатор файла
width - Ширина видео, заданная отправителем
height - Высота видео, заданная отправителем
duration - Продолжительность видео, заданная отправителем
thumb - Превью видео
mime_type - MIME файла, заданный отправителем
file_size - Размер файла

sendVideo - этот метод для отправки видео файлов, клиенты Telegram поддерживают видео в формате mp4 (другие форматы могут быть отправлены в виде документа).

chat_id - Уникальный идентификатор целевого чата или имя пользователя целевого канала
video - Видео для отправки. Вы можете либо передать file_id в виде строки для повторной отправки видео, которое уже находится на серверах Telegram, либо загрузить новый видеофайл, используя multipart/form-data.
duration - Продолжительность отправленного видео в секундах

width - Ширина видео
height - Высота видео
caption - Подпись к видео (также может использоваться при повторной отправке видео по file_id), 0-200 символов
disable_notification - Отправляет сообщение беззвучно. Пользователи iOS не получат уведомление, пользователи Android получат уведомление без звука.
reply_to_message_id -Если сообщение является ответом, идентификатор исходного сообщения
reply_markup - Дополнительные опции интерфейса. Объект, сериализованный в формате JSON для встроенной клавиатуры, пользовательской клавиатуры ответа, инструкций по скрытию клавиатуры ответа или принудительному получению ответа от пользователя

```
import requests

BASE_URL = 'https://api.telegram.org/bot'

TOKEN = '5580946066:AAH9SSwpAd_Pyp9pNkE9NrlD7DDZ53igeio'
ADMINS = [1865314469, ]
def pulling():
   count_message = 0
   while True:
       response = requests.get(f'{BASE_URL}{TOKEN}/getUpdates').json()
       if count_message != len(response['result']):
           count_message = len(response['result'])
           message = response['result'][-1]
           file_id = message['message']['video']['file_id']
           user_id = message['message']['from']['id']
           if user_id in ADMINS:

requests.get(f'{BASE_URL}{TOKEN}/sendVideo?chat_id={user_id}&video={file_id
}')

pulling()
```