Проект API. Знакомство с API

План урока

Аннотация

В уроке рассказывается о том, что такое API, почему и для чего создаётся API, как изучать новый API и работать с ним. Разбирается HTTP-API на базе StaticMapsAPI и Geocoder API. Изучается принцип работы с HTTP-API на языке Python.

На этом и последующих уроках для работы всех примеров необходимо подключение к сети Интернет.

1. Что такое API

Давайте представим себе, что мы с вами разработали какую-нибудь очень полезную программу. Например, электронную карту города. Программой пользуется всё большее количество людей. У нас, конечно же, есть канал обратной связи, по которому пользователи сообщают нам об ошибках в программе и каких функций им не хватает. Мы исправляем ошибки и стараемся расширять функциональность. Но силы наши конечны, а пожеланий с ростом популярности программы становится всё больше и больше. Делать всё мы не успеваем.
Да и нет смысла выполнять каждое пожелание, если мы понимаем, что данная функция нужна лишь очень небольшой аудитории. Но любым отказом мы расстраиваем наших пользователей, а это делать совсем не хочется. Как быть?

Вот одно из решений. Давайте вместо того, чтобы выполнять пожелания пользователей, дадим им инструмент, с помощью которого они сами смогут воплотить свои идеи. Например, можно написать библиотеку (вы ведь знакомы с библиотеками), в которой будут функции, если мы говорим про карты:

- нарисовать карту;
- нарисовать на ней точки, линии или какие-нибудь картинки;
- построить маршрут по карте;
- найти координаты какого-либо объекта и тому подобное.

Понятно, что таким образом мы дадим инструмент только программистам, но программиста можно нанять, и это уже не обязательно должен быть наш сотрудник. Если мы, помимо библиотеки, напишем инструкцию к ней, опишем её функции, как ими пользоваться и как построить приложение на основе нашей библиотеки, то сторонний программист вполне сможет разобраться с ней и решить задачу заказчика.

И вот так мы как бы строим мостик между нашей **очень полезной программой** и программистом, который хочет пользоваться ее возможностями. Мостик — это интерфейс. В англоязычной терминологии это называется *Application Programming Interface* или сокращенно **API** [эй-пи-ай].

API существует у большого количества программных продуктов: от операционной системы до интернет-сервисов. API — это обычный этап развития программных продуктов.

2. Зачем нужен API?

Как правило, API появляется, когда аудитория, использующая программный продукт, разрастается настолько, что своими силами команда-разработчик уже не успевает реализовывать все запросы пользователей.

Естественно, в коммерческом рабочем процессе ничто не делается просто так. Поэтому, создавая API, разработчики обязательно понимают, какие коммерческие цели они преследуют.

Можно просто предоставлять API за деньги: продавать разовую лицензию или брать плату за обращение к функциям. Часто API — это средство развития платформы. Например, *Windows API*. Все пишут программы под ОС Windows, в результате чего выигрывает вся платформа. API может решать и репутационные задачи — создавать лояльную пользовательскую аудиторию.

Встречаются и смешанные решения: одновременно существует бесплатное API с определёнными ограничениями (использование только в бесплатных открытых продуктах, с ограничением по количеству обращений) и его платная версия без таковых. На этом принципе построено большинство API в Яндексе.

Если это технически реально и логически осмысленно, то в интерфейс можно встраивать рекламу, которая добавляет заработок команде.

Полезно помнить, что API выпускается не просто так, и соотносить свои задачи с используемым инструментом. Особенно важно не забывать про возможные технические (по скорости работы, количеству запросов и т.п.) и юридические (лицензионные) ограничения использования того или иного API.

3. Варианты реализации API

Давайте посмотрим, как может выглядеть API. Один из вариантов — это библиотека, собранная под одну или несколько платформ. Она может либо содержать нужные нам функции или объекты в непосредственно в себе, либо использовать интернет для доступа к удалённому серверу с работающим на нем сервисом. Для того, чтобы пользоваться такой библиотекой, надо её получить (скачать, например), бесплатно или заплатив ее стоимость, установить и дальше применять, как и любую другую библиотеку. Иногда при оплате или регистрации к библиотеке прилагается уникальный ключ для того, чтобы она работала.

Ключ — это хитрая последовательность символов, для которой выполняется какое-то неизвестное нам, но известное авторам ключа условие. Случайно сгенерировать или подобрать ключ очень сложно. Обычно достаточно сложно для того, чтобы проще и дешевле было получить ключ легальным способом.

Есть вариант, при котором скачивать не требуется ничего. Это возможно, если сервис реализован как удалённый, использующий для доступа какой-либо стандартный протокол.

Например, HTTP. Обычно для доступа к такому сервису требуется иметь ключ и передавать его при каждом обращении к сервису. Ключ можно получить либо в автоматическом, либо в полуавтоматическом режиме, как правило, заполнив форму и выразив согласие с лицензионными соглашениями.

Протокол состоит из набора запросов и форматов ответов, предоставляемых сервисом.

HTTP-сервисы можно «пощупать» просто из браузера, поскольку протокол состоит из HTTP-запросов и ответов в форматах, поддерживаемых браузерами.

Пусть для нас изучение HTTP не является самоцелью, но остановиться на нём необходимо. Это тот инструмент, через который мы дальше будем взаимодействовать с различными API.

4. Немного о протоколе HTTP

Что же такое HTTP? Это протокол, позволяющий задавать запросы интернет-сервису и получать на них ответы.

В протоколе есть несколько разного вида запросов: GET, POST, HEAD, DELETE и т.д.

Как правило, запросы используются следующим образом:

- GET — для получения каких-либо данных с сервера;
- POST — для передачи большого объема данных на сервер и получения ответа;
- HEAD — для получения информации о данных с сервера;
- DELETE — для удаления данных с сервера.

Помимо вида запроса обычно требуется указать:

- адрес сервера, которому запрос направлен;
- путь к конкретной странице на сервере, которую необходимо получить, или к скрипту, который должен обработать запрос;
- дополнительные параметры запроса;
- вид ожидаемого ответа.

Протокол HTTP используется браузерами при просмотре интернет-страниц. По умолчанию применяется метод GET, формат ответа — html, json или какой-либо еще стандартный формат, и указывать это при каждом запросе не требуется. Остальная часть запроса записывается в виде:

```
http://static-maps.yandex.ru/1.x/?ll=37.677751,55.757718&spn=0.016457,0.00619&l=m
```

Здесь указаны:

- название протокола (**http**);
- адрес сервера, к которому мы обращаемся (**static-maps.yandex.ru**);
- путь к странице на этом сервере (до знака «?») (**/1.x/**);
- параметры вида ключ=значение, разделённые амперсандом (например, **ll=37.677751,55.757718**).

Запросы такого вида можно задавать в адресной строке браузера. Вставьте приведённую ссылку в адресную строку браузера и посмотрите на ответ.

Ответ состоит из заголовка, в котором указан формат ответа, код ошибки выполнения запроса и сам ответ. Вы наверняка слышали про коды состояния HTTP.

- 200 означает, что всё прошло успешно;
- 3xx — ошибки, связанные с перемещением страницы и т.п;
- 4xx — ошибки клиента (например, 403 — не хватает прав, 404 — файл не найден);
- 5xx — ошибки уровня приложения (запрос получен, а скрипт, который должен выдать ответ, ответа не выдал).

Если возвращаться к браузеру, то из всего этого пользователю показывается только содержимое ответа. При ошибке показывается код и расшифровка ошибки.

Про другие запросы, а также про коды ошибок, можно самостоятельно почитать на этой странице.

Кстати, среди кодов HTTP-ошибок есть код **418**. Узнайте на досуге, как он расшифровывается и что обозначает.

Но давайте от слов перейдём к делу и попробуем пощупать **Static API Яндекс.Карт**.

5. Знакомство с Yandex.Maps API

Вопрос: откуда нам знать про пути, ключи и их значения, из которых состоят запросы к API? Как понять, в каком формате придет ответ, и что он означает?

На все эти вопросы как раз и отвечает документация API. Страницу с документацией обычно можно обнаружить Поиском. Что же нужно там искать? Давайте посмотрим на содержание этого ресурса.

Здесь описываются состав, правила работы и протокол API Yandex.Maps.

HTTP-API состоит из трёх частей:

- StaticApi (получить картинку с картой);
- Геокодер (поиск топонимов, то есть картографических объектов);
- ППО (Поиск По Организациям).

Кроме того, в документации указано, что ключ для доступа **не нужен**. Лицензионное соглашение предписывает использование API в некоммерческих общедоступных **web-приложениях**.

Здесь необходимо оговориться, что в рамках данного курса мы рассматриваем серверное программирование. Для того, чтобы не нарушать лицензионное соглашение, необходимо, чтобы итоговое приложение имело и web-составляющую. Понятно, что во время разработки показывать результаты каждого запроса в web невозможно. Никто этого и не требует.
Но в итоговом приложении web-составляющая должна быть. Пока же мы только изучаем API и не создаём программных продуктов как таковых — это требование нас не затрагивает.
Но если в будущем вам предстоит им воспользоваться, учтите это требование сразу и потрудитесь его выполнить. Иначе придётся использовать платную версию API.

6. Yandex.Maps StaticApi

Документация находится на странице:
https://tech.yandex.ru/maps/doc/staticapi/1.x/dg/concepts/input_params-docpage/.

Здесь можно почитать про то, какие параметры могут быть в запросе, и что они означают.

Пример запроса:

```
https://static-maps.yandex.ru/1.x/?ll=37.677751,55.757718&spn=0.016457,0.00619&l=
```

В ответ придёт картинка с картой запрошенной области.

Посмотрите, что означают параметры **ll**, **spn**, **l** и поработайте с ними.

Тренировочное задание № 1: с помощью запросов к API через браузер получить:

1. Крупномасштабную схему с МГУ им. Ломоносова.
2. Спутниковый снимок Эйфелевой башни.
3. Спутниковый снимок Авачинского вулкана.
4. Спутниковый снимок космодрома Байконур.

Подсказка: для решения задачи можно открыть в браузере Я.Карты, найти объекты через поиск, щёлкнуть мышкой на карте и получить координаты для параметра **ll**. Параметр **spn** можно подобрать экспериментально.

7. Геокодер (поиск топонимических объектов)

Геокодер позволяет получать информацию об объектах, которые расположены на картах.

Мы предлагаем вам самостоятельно ознакомиться с документацией, попробовать сделать запросы, понять, что означают ответы геокодера, и ответить на вопрос второго тренировочного задания.

Тренировочное задание № 2: Получите координаты Якутска и Магадана в формате JSON. Выясните, какой город находится севернее: Якутск или Магадан?

8. Обращение к HTTP-сервису на языке Python

Поиграли? Замечательно. Теперь вспомните, что мы программируем на языке Python. Что же нужно для того, чтобы общаться с API Yandex.Карт из программы на Python?

Нужно написать программу, которая выполнит ровно те действия, которые вы только что совершили в браузере руками. Но как?

Нам потребуется библиотека для работы с протоколом http. Таких библиотек существует несколько. Все они позволяют передавать запросы и получать ответы от удалённых серверов. Рассмотрим, как это делается с помощью библиотеки requests.

Для работы с библиотекой нужно её импортировать:

```
import requests
```

Для выполнения запроса GET (вспоминаем протокол HTTP) используется функция **get()**.

```
response = requests.get(
    "http://geocode-maps.yandex.ru/1.x/?geocode=Якутск&format=json")
print(response, type(response))

----------------------------------------
```

```
<Response [200]> <class 'requests.models.Response'>
```

Если вместо «красивого» результата вы увидите много-много текста с исключениями, то, скорее всего, вы не смогли подключиться к серверу и получить ответ. В этом случае надо посмотреть на самую последнюю часть ответа и попытаться понять, в чём заключается проблема.

Функция **get()** возвращает объект класса **requests.models.Response**, который среди прочих содержит поля:

- status_code — код статуса (200 означает, что запрос выполнен успешно);
- reason — текстовая расшифровка статуса на английском языке (например, «Ok» или «Not Found»);
- content — ответ сервера.

У класса есть метод **bool()**, возвращающий True в случае успешного запроса и False в случае ошибки. Если произошла ошибка, то ответ сервера будет пустым. Таким образом, мы можем очень элегантно проверять успешность запроса.

Соберём полученные знания воедино.

Простейшая программа, выполняющая запрос к серверу, анализирующая код ответа и в случае успешного запроса печатающая полученную страницу, выглядит следующим образом:

```
import requests

# Готовим запрос.
geocoder_request = "http://geocode-maps.yandex.ru/1.x/?geocode=Якутск&format=jso

# Выполняем запрос.
response = None
try:
    response = requests.get(geocoder_request)
    if response:
        # Запрос успешно выполнен, печатаем полученные данные.
        print(response.content)
    else:
        # Произошла ошибка выполнения запроса. Обрабатываем http-статус.
        print("Ошибка выполнения запроса:")
        print(geocoder_request)
        print("Http статус:", response.status_code, "(", response.reason, ")")
except:
    print("Запрос не удалось выполнить. Проверьте подключение к сети Интернет.")

----------------------------------------
```

```
b'{"response":{"GeoObjectCollection":{"metaDataProperty":{
    "GeocoderResponseMetaData":{"request":"\xd0\xaf\xd0\xba\xd1\x83\xd1\x82\xd1\x
                               "found":"3","results":"10"}},
    "featureMember":[{"GeoObject":{"metaDataProperty":{
    "GeocoderMetaData":{"kind":"locality",
                        "text":"\xd0\xa0\xd0\xbe\xd1\x81\xd1\x81\xd0\xb8\xd1\x8f,
                        "precision":"other",
                        "Address":{"country_code":"RU","formatted":"\xd0\xa0\xd0\
                        "Components":...
```

Обратите внимание, что вся работа с сетью обёрнута в блок try..except. Это позволяет обрабатывать ошибки, которые находятся вне компетенции библиотеки requests. Попробуйте вытащить сетевой кабель из вашего компьютера или выключить wi-fi адаптер, а потом выполнить приведённый пример.

Что произошло? Почему?

Тренировочное задание № 3: Попробуйте «испортить» запрос (например, заменив 1.x на 1.x.1), чтобы увидеть, как обрабатывается ошибка запроса.

Если мы передадим в программу строку, содержащую правильно сформулированный запрос к API, то полученный ответ сможем дальше обработать в программе.

Например, из ответа геокодера сможем выбрать и отобразить отдельные поля так, чтобы ответ стал удобочитаемым для пользователя.

Поскольку json — это одним из популярнейших форматов обмена данными, у объекта **requests.Response** уже есть готовый метод **json()**, конструирующий json-объект из текста полученного ответа.

```
import requests

# Ищем город Якутск, ответ просим выдать в формате json.
geocoder_request = "http://geocode-maps.yandex.ru/1.x/?geocode=Якутск&format=jso

# Выполняем запрос.
response = None
try:
    response = requests.get(geocoder_request)
    if response:
        # Преобразуем ответ в json-объект
        json_response = response.json()

        # Получаем первый топоним из ответа геокодера.
        # Согласно описанию ответа, он находится по следующему пути:
        toponym = json_response["response"]["GeoObjectCollection"]["featureMember
```

```
        # Полный адрес топонима:
        toponym_address = toponym["metaDataProperty"]["GeocoderMetaData"]["text"]
        # Координаты центра топонима:
        toponym_coodrinates = toponym["Point"]["pos"]
        # Печатаем извлечённые из ответа поля:
        print(toponym_address, "имеет координаты:", toponym_coodrinates)
    else:
        print("Ошибка выполнения запроса:")
        print(geocoder_request)
        print("Http статус:", response.status_code, "(", response.reason, ")")
except:
    print("Запрос не удалось выполнить. Проверьте подключение к сети Интернет.")

----------------------------------------

Россия, Республика Саха (Якутия), Якутск имеет координаты: 129.732663 62.028103
```

Запросив изображение карты через StaticAPI, мы можем отрисовать его в окне программы.

Если посмотреть в браузере информацию о странице-картинке, полученной от StaticApi, то можно найти формат этой картинки. Это **PNG**. Для того, чтобы отобразить картинку на экране, запишем её в файл с расширением .png и отрисуем файл с помощью библиотеки pygame. Давайте посмотрим, как будет выглядеть такая программа.

```
import pygame
import requests
import sys
import os

response = None
try:
    map_request = "http://static-maps.yandex.ru/1.x/?ll=37.530887,55.703118&spn=0
    response = requests.get(map_request)

    if not response:
        print("Ошибка выполнения запроса:")
        print(geocoder_request)
        print("Http статус:", response.status_code, "(", response.reason, ")")
        sys.exit(1)
except:
    print("Запрос не удалось выполнить. Проверьте наличие сети Интернет.")
    sys.exit(1)

# Запишем полученное изображение в файл.
map_file = "map.png"
```

```
try:
    with open(map_file, "wb") as file:
        file.write(response.content)
except IOError as ex:
    print("Ошибка записи временного файла:", ex)
    sys.exit(2)

# Инициализируем pygame
pygame.init()
screen = pygame.display.set_mode((600, 450))
# Рисуем картинку, загружаемую из только что созданного файла.
screen.blit(pygame.image.load(map_file), (0, 0))
# Переключаем экран и ждем закрытия окна.
pygame.display.flip()
while pygame.event.wait().type != pygame.QUIT:
    pass
pygame.quit()

# Удаляем за собой файл с изображением.
os.remove(map_file)
```

В примере нам встретилась функция **sys.exit()**. Как следует из её имени, она прекращает работу программы и возвращает результатом значение, переданное в неё в качестве параметра.

Выполните пример. Попробуйте задать другие параметры запроса. Подумайте, можно ли обойтись без сохранения полученного контента в файл, а сразу передать его в окно pygame?