

# Знакомство с API на примере Django REST framework



Александр  
Бардин



# Александр Бардин

Python-разработчик в Open Solutions



---

# План занятия

1. [Как именуются ресурсы в REST API](#)
2. [Установка и настройка DRF](#)
3. [View](#)
4. [Роутер](#)
5. [Сериализаторы](#)
6. [ViewSet](#)
7. [Как дебажить DRF](#)

---

# Повторение



Что такое API?



# Повторение

**API** — интерфейс для программного взаимодействия. Поэтому API должен содержать ответ в четко заданной структуре, например, в JSON-формате с фиксированным списком полей.

API позволяет программам общаться друг с другом.

**Пример:** мобильное приложение взаимодействует с сервером по API поверх HTTP.

API является самым популярным способом общения между бэкендом и фронтендом.

---

# Повторение



Что такое REST API?



# REST API

**REST API** — архитектурный стиль проектирования API.

Основные требования:

- взаимодействие клиент-сервер
- запросы содержат в себе все необходимое состояние
- строгое именование ресурсов
- использование семантики HTTP-методов и определенных кодов возврата

# Повторение

<b>SAFE METHODS</b> NO ACTION ON SERVER	<b>GET</b>	HTTP/1.1 MUST IMPLEMENT THIS METHOD
	<b>HEAD</b>	<b>INSPECT</b> RESOURCE HEADERS
<b>MESSAGE WITH BODY</b> SEND DATA TO SERVER	<b>PUT</b>	<b>DEPOSIT</b> DATA ON SERVER — INVERSE OF GET
	<b>POST</b>	<b>SEND</b> INPUT DATA FOR PROCESSING
	<b>PATCH</b>	<b>PARTIALLY MODIFY</b> A RESOURCE
	<b>TRACE</b>	<b>ECHO</b> BACK RECEIVED MESSAGE
	<b>OPTIONS</b>	SERVER <b>CAPABILITIES</b>
	<b>DELETE</b>	DELETE A RESOURCE — NOT GUARANTEED





# Как именуются ресурсы в REST API



# Как именуются ресурсы в REST API

Примеры заказов:

GET /api/orders/

GET /api/orders/1/

POST /api/orders/

PATCH /api/orders/1/

PUT /api/orders/1/

DELETE /api/orders/1/

---

# Коды возврата



Какие коды из 200х должны возвращаться на каждый метод в REST API:

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes#2xx\\_Success](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#2xx_Success) ?

## Методы:

- получение сущности (GET)
- создание сущности (POST)
- обновление сущности (PUT)
- частичное обновление сущности (PATCH)
- удаление сущности (DELETE)

---

# Коды ответов в REST API



Какие коды из 200х должны возвращаться на каждый метод в REST API:

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes#2xx\\_Success](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#2xx_Success) ?

Проверьте себя:

- **POST** возвращает 201;
- **DELETE** — 204 или 200;
- **GET / PATCH / PUT** — 200.



# **Установка и настройка DRF**



# Установка и настройка DRF

DRF – является стандартом для реализации API на Django.

Ссылка на DRF: <https://www.django-rest-framework.org/>

---

# Компоненты DRF

DRF состоит из следующих основных компонент:

- View и viewset
- Сериализаторы
- Роутер
- Система управления доступом
- Фильтры



# Демонстрация

- установка DFR,
- указание параметров в settings,
- написание одного view,
- просмотр ответа.





# View

# View

## Class-based view

Методы класса описывают обработчиков для HTTP-методов

## ViewSet

Каждый метод класса описывает обработку соответствующего запроса на ресурс:

- создание,
- получение,
- список,
- обновление,
- удаление.

Если какой-то метод не реализован, то Django будет возвращать **ошибку 405**.



# Роутер

# Роутер

В роутере описываются подключаемые view.  
Роутер мапит HTTP-методы на методы view.

```
from django.urls import path, include
from rest_framework import routers

from orders import views as order_views

router = routers.DefaultRouter()
router.register("orders", order_views.OrderViewSet)

urlpatterns = [
    path('api/v1/', include(router.urls)),
    path('admin/', admin.site.urls),
]
```



# Сериализаторы



# Сериализаторы

Когда ресурс заведен поверх модели, возникает вопрос – как нам сконвертировать её в JSON?



# Сериализаторы

В сериализаторе мы описываем представление модели, указываем поля, которые мы хотим возвращать, и их типы. Для того чтобы сэкономить время, можно использовать **ModelSerializer**, который самостоятельно выводит типы на основании указанной в Meta модели.



# Демонстрация

- создание сериализатора,
- создание JsonSerializer,
- сравним преимущества и недостатки подходов.





# ViewSet



# ViewSet

ViewSet является встроенной в DRF батареей для стандартной CRUD (create-read-update-delete) логики на моделями.

Конфигурация Viewset описывается с помощью атрибутов и методов класса.

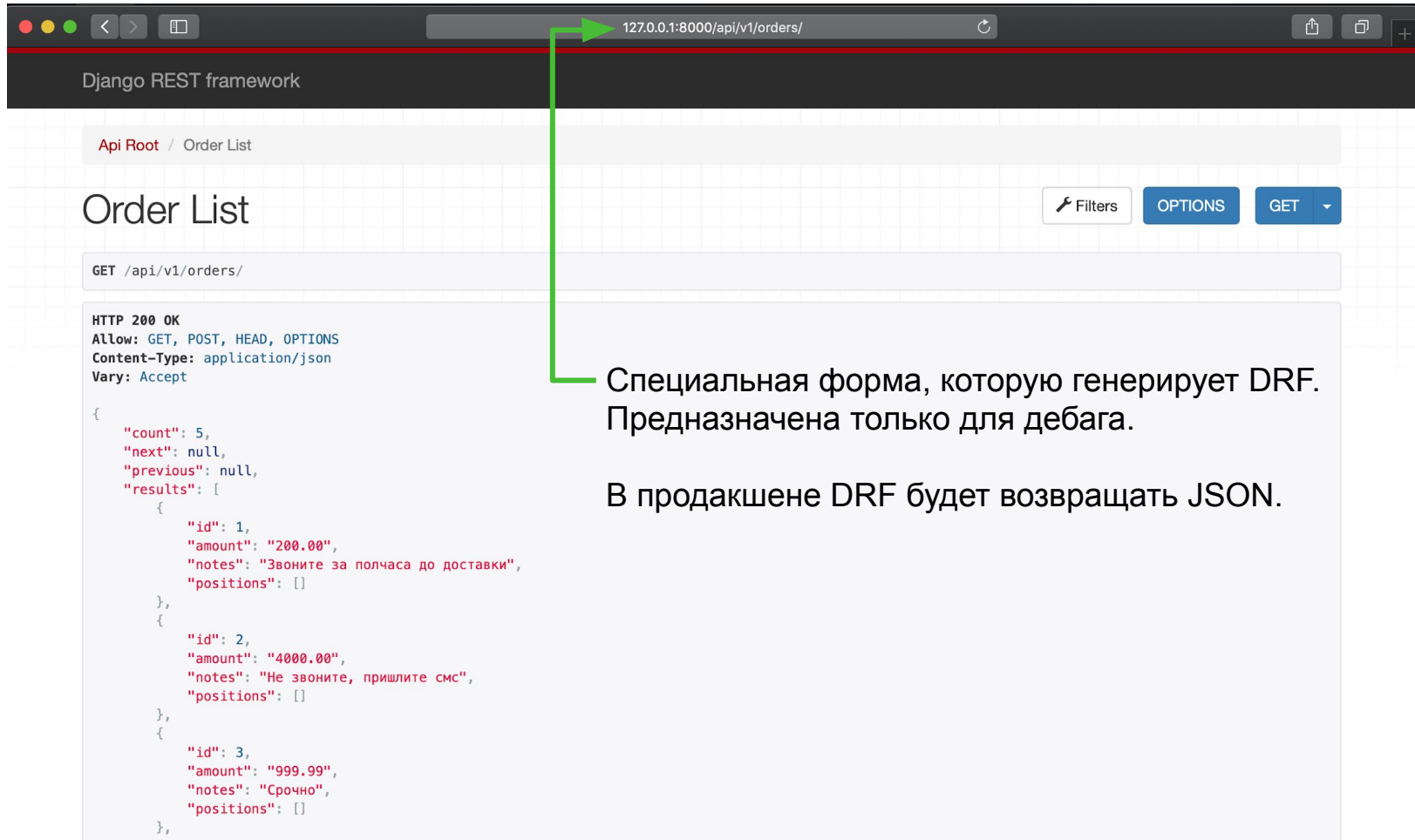
Мы описываем, что мы достаём и в каком формате отдаём, с помощью атрибутов **queryset** и **serializer**.



# Демонстрация

Перепишем пример на ViewSet и оценим количество сокращенного кода.

# Результат



127.0.0.1:8000/api/v1/orders/

Django REST framework

Api Root / Order List

## Order List

GET /api/v1/orders/

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "count": 5,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "amount": "200.00",
      "notes": "Звоните за полчаса до доставки",
      "positions": []
    },
    {
      "id": 2,
      "amount": "4000.00",
      "notes": "Не звоните, пришлите смс",
      "positions": []
    },
    {
      "id": 3,
      "amount": "999.99",
      "notes": "Срочно",
      "positions": []
    }
  ]
}
```

Специальная форма, которую генерирует DRF. Предназначена только для дебага.

В продакшене DRF будет возвращать JSON.

---

# Как дебажить DRF

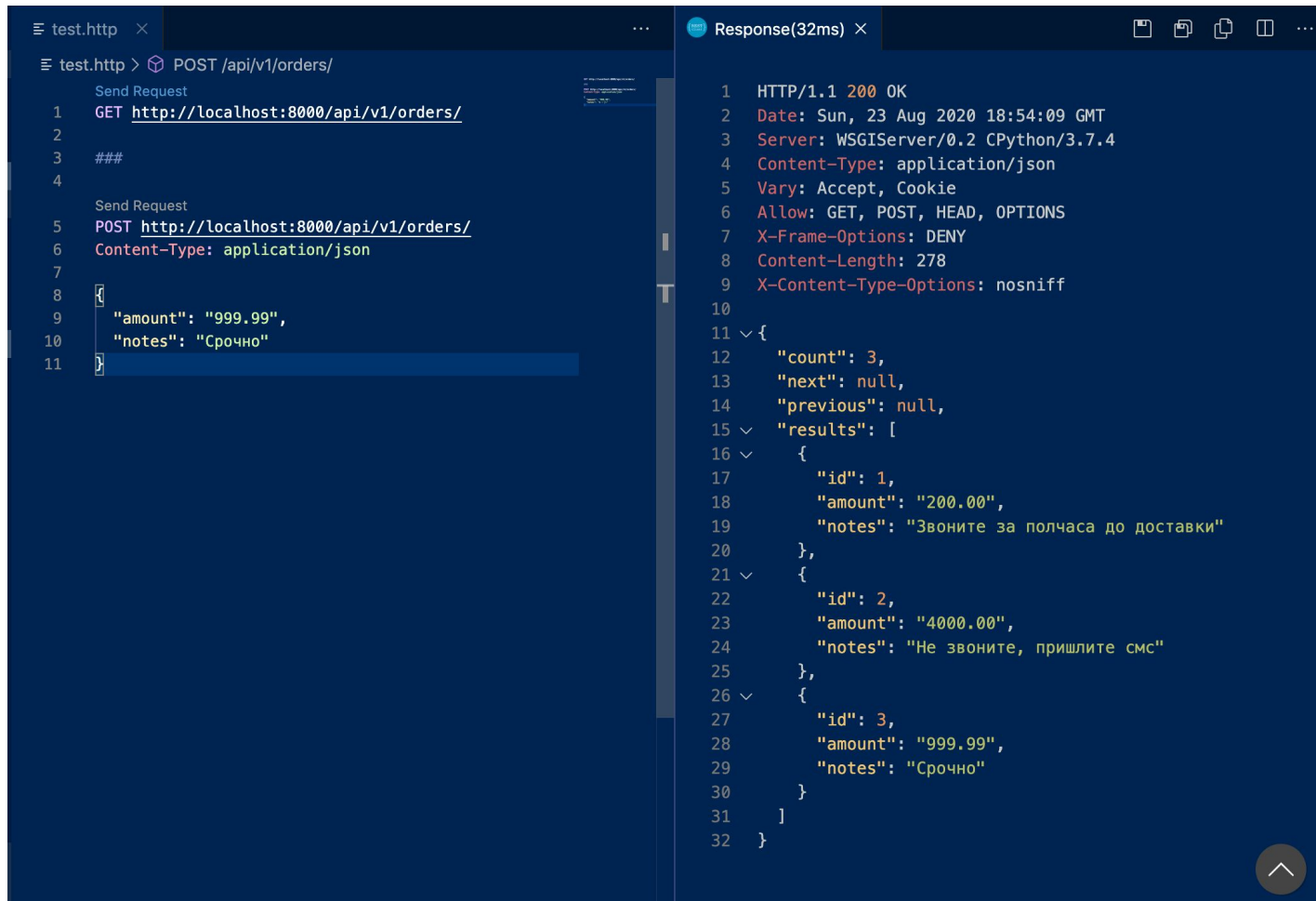
# Как дебажить DFR

Для дебаггинга DFR можно использовать HTTP-клиенты:

- Postman,
- HTTP Client в Visual Studio Code,
- HTTP клиент в Pycharm (только в платной версии).

Также можно использовать стандартный DRF-интерфейс, который доступен только при запуске в Debug-версии.

# Как дебажить DFR: http client в VSCode



The image shows a screenshot of the VS Code interface with two panels. The left panel, titled 'test.http', contains an HTTP client request. The right panel, titled 'Response(32ms)', shows the JSON response to the request.

```
test.http > POST /api/v1/orders/
Send Request
1 GET http://localhost:8000/api/v1/orders/
2
3 ###
4
Send Request
5 POST http://localhost:8000/api/v1/orders/
6 Content-Type: application/json
7
8 {
9   "amount": "999.99",
10  "notes": "Срочно"
11 }
```

```
Response(32ms)
1 HTTP/1.1 200 OK
2 Date: Sun, 23 Aug 2020 18:54:09 GMT
3 Server: WSGIServer/0.2 CPython/3.7.4
4 Content-Type: application/json
5 Vary: Accept, Cookie
6 Allow: GET, POST, HEAD, OPTIONS
7 X-Frame-Options: DENY
8 Content-Length: 278
9 X-Content-Type-Options: nosniff
10
11 {
12   "count": 3,
13   "next": null,
14   "previous": null,
15   "results": [
16     {
17       "id": 1,
18       "amount": "200.00",
19       "notes": "Звоните за полчаса до доставки"
20     },
21     {
22       "id": 2,
23       "amount": "4000.00",
24       "notes": "Не звоните, пришлите смс"
25     },
26     {
27       "id": 3,
28       "amount": "999.99",
29       "notes": "Срочно"
30     }
31   ]
32 }
```

---

# Итоги

Сегодня на занятии мы:

- Повторили, что такое REST API.
- Установили DjangoRestFramework.
- Узнали, как дебажить проект с API.





# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

 [Александр Бардин](#)

**Александр Бардин**