

Django Essential

GraphQL django api (часть 2)

Django Essential

Introduction



Лазорык Михаил

Software developer, 3 года опыта

 mykhailo.lazoryk

 mykhailo-lazoryk



Django Essential

Тема урока

GraphQL django api (часть 2)

Django Essential

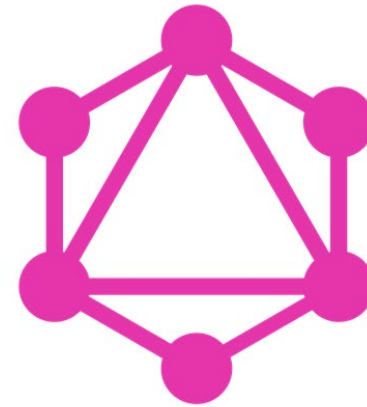
GraphQL django api (часть 2)

1. Создание схемы
2. Создание запроса (Query)
3. Создание мутаций
4. Тестирование GraphQL на практике

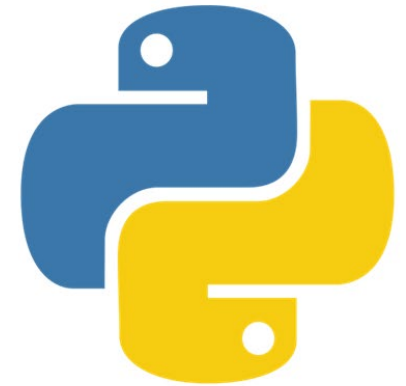
Django Essential

Создание схемы

- Установка библиотеки `graphene_django`
- Создание файла `schema.py` в корне приложения
- Создание `объекта` схемы
- Создание файла `schema.py` в корне проекта
- Добавление `graphene_django` в `INSTALLED_APPS`
- Добавление переменной `GRAPHENE` в настройки
- Добавление `url`



GraphQL



pythonTM

Django Essential

Создание запроса (Query)

- Для определения типов в GraphQL используем `DjangoObjectType`, `ObjectType`
- Чтобы создать Query используем `ObjectType`
- В Query необходимо добавить `Resolvers` это даст возможность GraphQL найти адрес где искать запрашиваемую информацию

```
class Query(ObjectType):
    dishwasher = graphene.Field(DishwasherType, id=graphene.Int())
    tv = graphene.Field(TVType, id=graphene.Int())
    dishwashers = graphene.List(DishwasherType)
    tvs = graphene.List(TVType)

    def resolve_dishwasher(self, info, **kwargs):
        id = kwargs.get('id')
        if id is not None:
            return Dishwasher.objects.get(pk=id)
        return None

    def resolve_tv(self, info, **kwargs):
        id = kwargs.get('id')
        if id is not None:
            return TV.objects.get(pk=id)
        return None

    def resolve_dishwashers(self, info, **kwargs):
        return Dishwasher.objects.all()

    def resolve_tvs(self, info, **kwargs):
        return TV.objects.all()
```

Django Essential

Создание мутаций

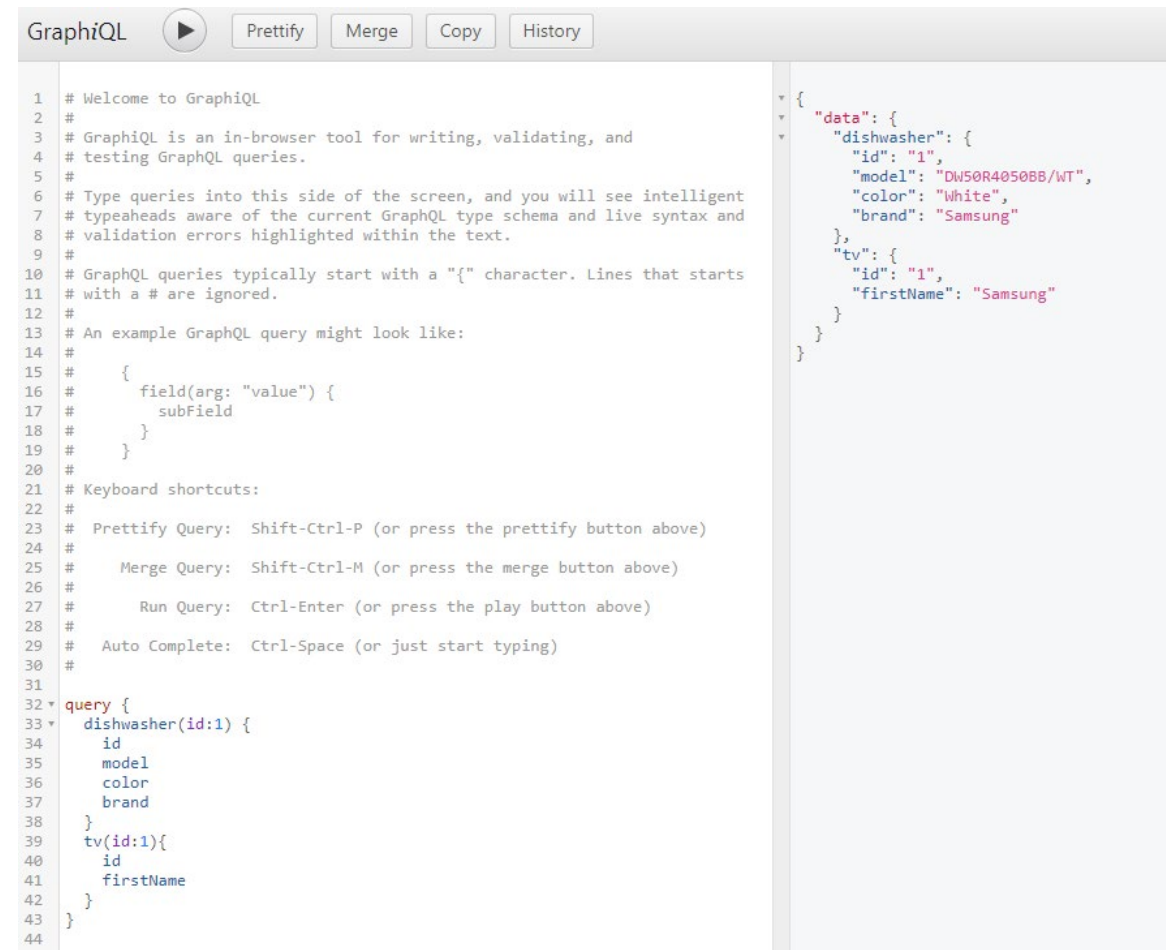
- `InputObjectType` служит для определения того, какие поля можно использовать для изменения с помощью GraphQL.
- Для создания мутаций используем класс `graphene.Mutation`. При создании мутации нужно **наследоваться** от этого класса.
- Для работы мутаций нужно внутри класса определить метод `mutate` и в нем прописывать логику для **создания** или **изменения** объекта в базе.
- С помощью мутаций можно как **создавать** так и **обновлять** данные, но для этого рекомендуется использовать **разные классы**.
- Последний штрих с мутациями - это создать **тип мутаций** (class Mutation).
- Чтобы подключить мутации к схеме используем `graphene.Schema`.

Django Essential

Тестирование GraphQL на практике

Для тестирования используем ранее прописанный линк в urls -> <http://127.0.0.1:8000/graphql>.

В результате откроется следующая страница.



The screenshot shows the GraphQL Playground interface. On the left, there is a text editor with a GraphQL query. The query is as follows:

```
1 # Welcome to GraphQL
2 #
3 # GraphQL is an in-browser tool for writing, validating, and
4 # testing GraphQL queries.
5 #
6 # Type queries into this side of the screen, and you will see intelligent
7 # typeahead aware of the current GraphQL type schema and live syntax and
8 # validation errors highlighted within the text.
9 #
10 # GraphQL queries typically start with a "{" character. Lines that starts
11 # with a # are ignored.
12 #
13 # An example GraphQL query might look like:
14 #
15 # {
16 #   field(arg: "value") {
17 #     subField
18 #   }
19 # }
20 #
21 # Keyboard shortcuts:
22 #
23 # Prettify Query: Shift-Ctrl-P (or press the prettify button above)
24 #
25 # Merge Query: Shift-Ctrl-M (or press the merge button above)
26 #
27 # Run Query: Ctrl-Enter (or press the play button above)
28 #
29 # Auto Complete: Ctrl-Space (or just start typing)
30 #
31
32 query {
33   dishwasher(id:1) {
34     id
35     model
36     color
37     brand
38   }
39   tv(id:1){
40     id
41     firstName
42   }
43 }
44
```

On the right, the JSON response is displayed:

```
{
  "data": {
    "dishwasher": {
      "id": "1",
      "model": "DW50R4050BB/WT",
      "color": "White",
      "brand": "Samsung"
    },
    "tv": {
      "id": "1",
      "firstName": "Samsung"
    }
  }
}
```


Django Essential

GraphQL django api (часть 2)

1. Создание схемы
2. Создание запроса (Query)
3. Создание мутаций
4. Тестирование GraphQL на практике

Проверка знаний

TestProvider.com



Проверьте как Вы усвоили данный материал на TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Django Essential

Спасибо за внимание! До новых встреч!



Лазорык Михаил
Software developer



Информационный видеосервис для разработчиков программного обеспечения

