Django: пути и view-функции | Я.Шпора

Протоколы и типы запросов

Сервер и клиент обмениваются информацией по протоколу **HTTP** (данные передаются без шифрования) или по протоколу **HTTPS** (при передаче данные шифруются).

HTTP-запросы могут быть разного типа.

GET-запросы применяют, чтобы получить информацию от сервера. При GET-запросе все параметры запроса передаются в URL.

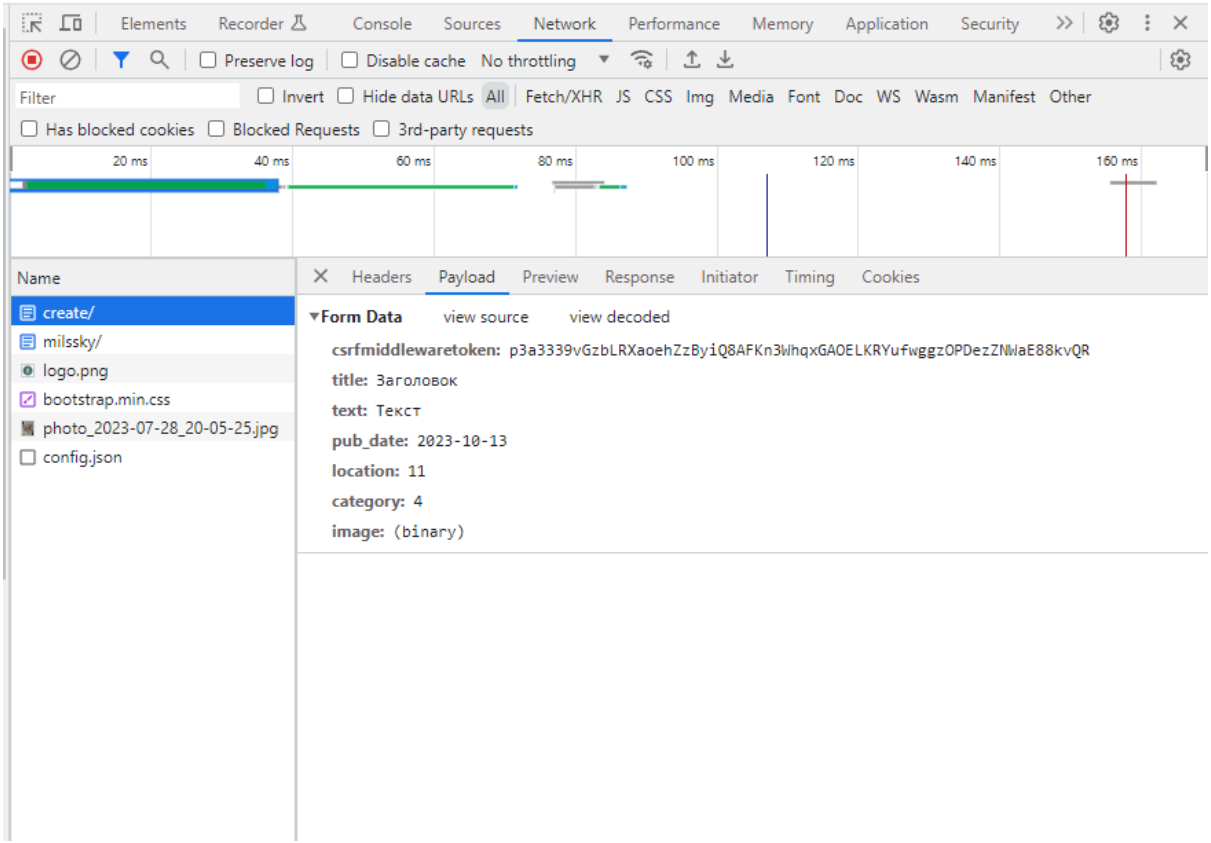
```
https://ya.ru/search/?text=практикум
```

POST-запросы обычно применяют, чтобы сохранить или изменить данные на сервере. При POST-запросе информация передаётся **в теле запроса**.

Увидеть HTTP-запросы и ответы сервера можно в панели *Developer Tools (Инструменты разработчика)* в браузере.

- Чтобы открыть инструменты разработчика в Windows — нажмите **F12** (**Fn+F12**).
- Чтобы открыть инструменты разработчика в macOS — нажмите **Ctrl+Shift+I** или **Cmd+Option+I**.

На скриншоте — панель Developer Tools, в ней видны данные, которые переданы в теле POST-запроса.



Адреса (URL)

Абсолютный адрес — это полный адрес веб-ресурса, с указанием протокола и домена; в общем виде он выглядит так:

```
<схема или протокол>://<ip адрес или домен сервера>/<путь>?<параметры>
Например:
https://practicum.yandex.ru/catalog/programming/?searchText=python
```

Относительный адрес — это адрес в пределах определённого домена: адрес берётся **относительно** того ip-адреса или домена, с которого отправлен запрос.

```
Вместо абсолютного адреса
https://practicum.yandex.ru/catalog/programming/?searchText=python
применяем относительный адрес:
/catalog/programming/?searchText=python
```

Маршрутизатор Django

В Django адрес запроса и функция-обработчик для этого адреса связываются с помощью функции `path()`. Эта функция принимает обязательные параметры:

- **route** — относительный адрес или шаблон адреса: строка-образец, с которой сравнивается полученный запрос.
- **view** — функция-обработчик для запрошенного адреса: если запрошенный URL соответствует **route**, вызов будет перенаправлен в указанную view-функцию.

В Django перечень путей `path()` хранят в списке `urlpatterns` в файле *urls.py*.

```
# Файл urls.py
from django.urls import path

...

urlpatterns = [
    path('<route - относительный_адрес>', <view - имя_функции_обработчика
    ...,
    ...,
]
...

# Пример
# path('catalog/', views.product_list)
```

Третий, необязательный параметр для функции `path()` — имя пути `name`. В коде проекта это имя можно применять вместо конкретных адресов.

```
# catalog/urls.py

urlpatterns = [
    # Имя для path лучше задавать такое же, как и имя функции.
    path('catalog/', views.product_list, name='product_list'),
    path('catalog/<int:pk>/', views.product_detail, name='product_detail'
]
```

Запросы к адресам, которых нет в *urls.py*, вернут ошибку *404 Page Not Found, «страница не найдена»*.

Для однотипных адресов можно применять **переменные** и **конвертеры пути**. Для этого в шаблоне адреса ожидаемый тип и имя переменной заключают в угловые скобки `<>`.

```
urlpatterns = [
    path('catalog/<int:pk>/', views.product_detail),
]
# Общий синтаксис для применения конвертеров и переменных:
# path('catalog/<конвертер_пути:переменная>/', имя_функции_обработчика)

# Запросы к адресам вида catalog/1/, catalog/65/, catalog/212/ и к другим
# будут обработаны view-функцией product_detail().

# А запрос к адресу catalog/main/ не будет передан на обработку в product_
# он не соответствует пути 'catalog/<int:pk>/',
# ведь "main" не конвертируется в целое положительное число.
```

Популярные конвертеры:

- `str` — ожидает любую непустую строку, исключая разделитель пути `'/'`.

  Если в шаблоне пути конвертер не указан явно, то по умолчанию будет применён именно `str`: шаблон адреса `<username>/` идентичен шаблону `<str:username>/`.
- `int` — ожидает ноль или любое целое положительное число: `<int:pk>/`.
- `slug` ожидает строку из латинских букв, цифр и символов `-` и `_`: `<slug:category_slug>/`.

Файлы *urls.py* из приложений подключаются к корневому файлу *urls.py* с помощью функции `include()`.

```
# название_проекта/urls.py (корневой urls.py проекта)

from django.urls import include, path

urlpatterns = [
```

```
    path('catalog/', include('catalog.urls')),
]
```

View-функции

View-функции принимают на вход экземпляр класса **HttpRequest** (объект запроса) и возвращают экземпляр класса **HttpResponse** (объект ответа).

Во view-функцию можно передать дополнительные аргументы — переменные из шаблона адреса в `path()`.

```
# urls.py
urlpatterns = [
    path('catalog/<int:pk>/', views.product_detail),
]

# views.py
def product_detail(request, pk):
    return HttpResponse(f'Запрошен продукт с номером {pk}')
```

Имя переменной в `path` и имя параметра view-функции должны совпадать:

```
# urls.py
urlpatterns = [
    path('catalog/<int:pk>/', views.product_detail),
]

# views.py
def product_detail(request, pk):
    return HttpResponse(f'Запрошен продукт с номером {pk}')
```

А такой вариант не сработает:

```
# В path() переменная названа id, а во view-функции - pk.
# Так нельзя: имена должны быть одинаковыми.

# urls.py
urlpatterns = [
    path('catalog/<int:id>/', views.product_detail),
```

```
]

# views.py
def product_detail(request, pk):
    return HttpResponse(f'Запрошен продукт с номером {pk}')
```

Возникнет ошибка:

```
TypeError at /catalog/57/
post_detail() got an unexpected keyword argument 'id'
```