

# Серверная разработка ПО

## Лекция 5

## Определение параметров через функцию `re_path()`

Вернемся к нашему приложению `firstapp`, которое мы создали в предыдущей лекции, и определим в этом приложении в файле `views.py` следующие дополнительные функции:

```
def products(request, productid):  
    output = "<h2>Продукт № {0}</h2>".format(productid)  
    return HttpResponse(output)  
  
def users(request, id, name):  
    output = "<h2>Пользователь</h2><h3> id: {0} Имя: {1}</h3>".format(id, name)  
    return HttpResponse(output)
```

Здесь функция `products` кроме параметра `request` принимает также параметр `productid` (идентификатор товара). Отправляемый пользователю ответ содержит значение этого параметра. Функция `users` принимает два дополнительных параметра: `id` и `name` (идентификатор и имя пользователя). И подобным же образом эти данные отправляются обратно пользователю.

Теперь изменим файл `urls.py`, чтобы он мог сопоставить эти функции с запросами пользователя:

```
re_path(r'^products/(?<productid>\d+)/', views.products),  
re_path(r'^users/(?P<id>\d+)/(?P<name>\D+)', views.users),
```

Для представления параметра в шаблоне адреса используется выражение `?P<>`. Общее определение параметра соответствует формату:

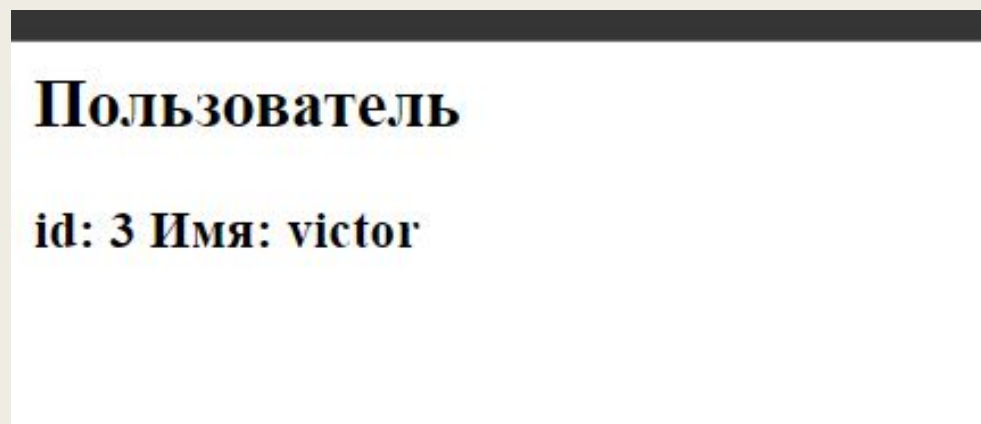
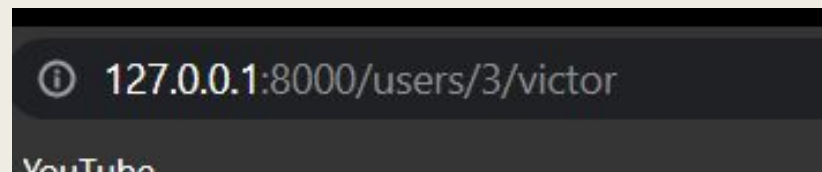
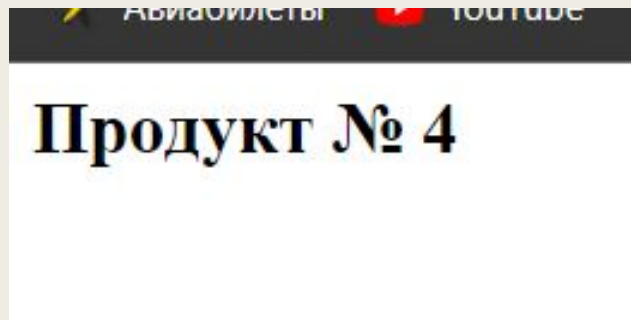
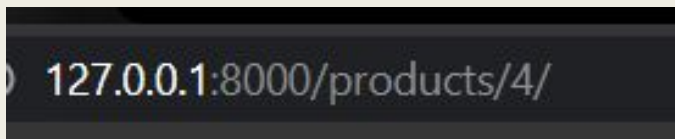
`(?P<имя_параметра>регулярное_выражение)`

В угловых скобках помещается название параметра. После закрывающей угловой скобки следует регулярное выражение, которому должно соответствовать значение параметра.

Например, фрагмент: `?P<productid>\d+` определяет, что параметр называется `productid`, и он должен соответствовать регулярному выражению `\d+`, т. е. представлять последовательность цифр.

Во втором шаблоне адреса: `(?P<id>\d+)/(?P<name>\D+)` определяются два параметра: `id` и `name`. При этом параметр `id` должен представлять число, а параметр `name` - состоять только из буквенных символов. Ну и также отметим, что количество и названия параметров в шаблонах адресов URL соответствуют количеству и названиям параметров соответствующих функций, которые обрабатывают запросы по этим адресам.

Теперь мы можем через адресную строку передать от пользователя данные в приложение в виде запроса. Наберем в адресной строке браузера следующий текст:



Однако если мы в запросе не передадим значение для параметра или передадим значение, которое не соответствует регулярному выражению, то система не сможет найти ресурс для обработки такого запроса и выдаст сообщение об ошибке. Например если пользователь запросит информацию о продукте, но при этом не укажет идентификационный номер продукта, то система выдаст в ответ сообщение об ошибке.

Для предотвращения ошибок такого рода можно в функции `products`, расположенной в файле `views.py`, определить значение параметра по умолчанию

```
def products(request, productid=1):  
    output = "<h2>Продукт № {0}</h2>".format(productid)  
    return HttpResponse(output)
```

То есть если в функцию не было передано значение для параметра `productid`, то он получает значение по умолчанию 1. В этом случае в файле `urls.py` надо дополнительно определить еще один маршрут:

```
urlpatterns = [  
    path('', views.index),  
    re_path(r'^about', views.about),  
    re_path(r'^contact', views.contact),  
    re_path(r'^products/(?P<productid>\d+)/', views.products),  
    re_path(r'^users/(?P<id>\d+)/(?P<name>\D+)', views.users),  
    re_path(r'^products', views.products),  
    path('admin/', admin.site.urls),  
]
```

## Определение параметров через функцию

### path()

Параметры функции path () заключаются в угловые скобки в формате:

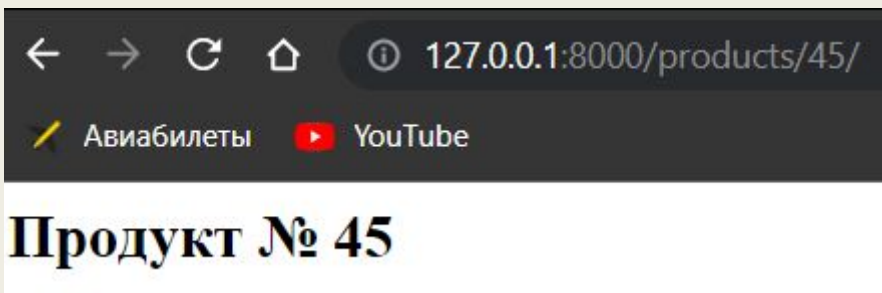
<спецификатор: название\_параметра>

Вернемся к функциям, описанным в файле

```
def products(request, productid=1):  
    output = "<h2>Продукт № {0}</h2>".format(productid)  
    return HttpResponse(output)  
  
def users(request, id, name):  
    output = "<h2>Пользователь</h2><h3> id: {0} Имя: {1}</h3>".format(id, name)  
    return HttpResponse(output)
```

Определим для них параметры в файле urls.py с помощью функции path ():

```
urlpatterns = [  
    path('', views.index),  
    re_path(r'^about', views.about),  
    re_path(r'^contact', views.contact),  
    path('products/<int:productid>/', views.products),  
    path('users/<int:id>/name/', views.users),  
    re_path(r'^products', views.products),  
    path('admin/', admin.site.urls),  
]
```



Как можно видеть, функция `path()` работает здесь аналогично функции `re_path()`. То есть в зависимости от предпочтений программиста можно использовать любую из этих функций.

В нашем примере в маршруте обращения к странице с продуктами параметр `productid` имеет спецификатор `int` (целое число).

По умолчанию Django предоставляет следующие спецификаторы параметров функции:

- `str` - соответствует любой строке за исключением символа `(/)`. Если спецификатор не указан, то используется по умолчанию;
- `int` - соответствует любому положительному целому числу;
- `slug` - соответствует последовательности буквенных символов ASCII, цифр, дефиса и символа подчеркивания, например: `building-your-1st-django-site`;
- `uuid` - соответствует идентификатору UUID, например: `075194d3-6885-417e-a8a8-6c931e272f00`;
- `path` - соответствует любой строке, которая также может включать символ `(/)`, в отличие от спецификатора `str`.



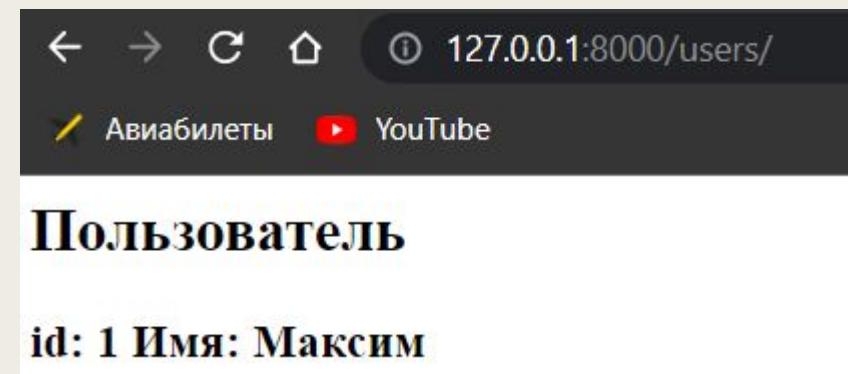
## Определение параметров по умолчанию в функции path()

Для примера зададим для функций в файле views.py значения параметров по умолчанию для тех страниц сайта, которые выдают информацию о продуктах и пользователях. Мы уже ранее задавали для продуктов значение по умолчанию: productid = 1. Теперь зададим для пользователя значения по умолчанию идентификатора пользователя (id=1) и имени пользователя (name="Максим").

```
def users(request, id = 1, name = "Максим"):
    output = "<h2>Пользователь</h2><h3> id: {0} Имя: {1}</h3>".format(id, name)
    return HttpResponse(output)
```

После этого для функций products и users в файле urls.py надо определить по два маршрута.

```
urlpatterns = [
    path('', views.index),
    re_path(r'^about', views.about),
    re_path(r'^contact', views.contact),
    path('products/', views.products),
    path('products/<int:productid>', views.products),
    path('users/', views.users),
    path('users/<int:id>/name/', views.users),
    path('admin/', admin.site.urls),
]
```





## Параметры строки запроса пользователя

Следует четко различать параметры, которые передаются через интернет-адрес (URL), и параметры, которые передаются через строку запроса. Например, в запросе:

`http://localhost/index/3/Виктор/`

два последних сегмента: `3/Виктор/` представляют собой параметры URL. А в запросе:

`http://localhost/index?id=3&name= Виктор`

те же самые значения `3` и `Виктор` представляют собой параметры строки запроса.

Параметры строки запроса указываются после символа вопросительного знака (?). Каждый такой параметр представляет собой пару «ключ-значение»). Например, в параметре `id=3`: `id` - это ключ параметра, а `3` - его значение. Параметры в строке запроса отделяются друг от друга знаком амперсанда (&).

Для получения параметров из строки запроса применяется метод `request.GET.get ()`

```
def products(request, productid=1):  
    category = request.GET.get("cat", "")  
    output = "<h2>Продукт № {0} Категория: {1} </h2>".format(productid, category)  
    return HttpResponse(output)  
  
def users(request):  
    id = request.GET.get("id", 1)  
    name = request.GET.get("name", "Максим")  
    output = "<h2>Пользователь</h2><h3> id: {0} Имя: {1}</h3>".format(id, name)  
    return HttpResponse(output)
```

Функция `products` принимает обычный параметр `productid` (идентификатор продукта), который будет передаваться через интернет-адрес (URL). И также из строки запроса извлекается значение параметра `cat` (категория продукта) - `request.GET.get("cat", "")`. Здесь первый аргумент функции - это название параметра строки запроса, значение которого надо извлечь, а второй аргумент - значение по умолчанию (на случай, если в строке запроса не оказалось подобного параметра). В функции `users` из строки запроса извлекаются значения параметров `id` и `name`. При этом заданы следующие значения параметров по умолчанию: `id=1`, `name="Максим"`.

При обращении к приложению по адресу:

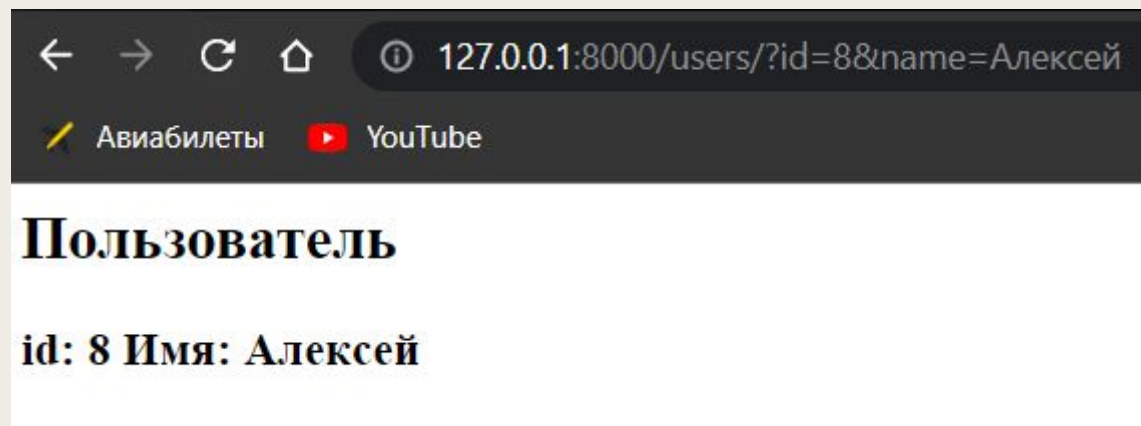
`http://127.0.0.1:8000/products/3/?cat=Телефоны`

число 3 будет представлять параметр URL, присваиваемый параметру `productid`, а значение `cat=Телефоны` - представлять параметр `cat` строки запроса.

А при обращении по адресу:

`http://127.0.0.1:8000/users/?id=8&name=Алексей`

значения 8 и Алексей будут представлять соответственно параметры `id` и `name`



# Переадресац

## ия

При перемещении документа с одного адреса на другой мы можем воспользоваться механизмом переадресации, чтобы указать пользователям и поисковику, что документ теперь доступен по новому адресу. Переадресация бывает **временная** и **постоянная**.

При временной переадресации мы указываем, что документ временно перемещен на новый адрес. В этом случае в ответ отправляется статусный код 302.

При постоянной переадресации мы уведомляем систему, что документ теперь постоянно будет доступен по новому адресу.

Для создания временной переадресации применяется класс: `HttpResponseRedirect` Для создания постоянной переадресации применяется класс: `HttpResponsePermanentRedirect` Оба класса расположены в пакете `django.http`.

```
from django.shortcuts import render
from django.http import HttpResponseRedirect, HttpResponseRedirectPermanentRedirect

def index(request):
    return HttpResponse("<h2>Главная</h2>")

def about(request):
    return HttpResponse("<h2>0 сайте</h2>")

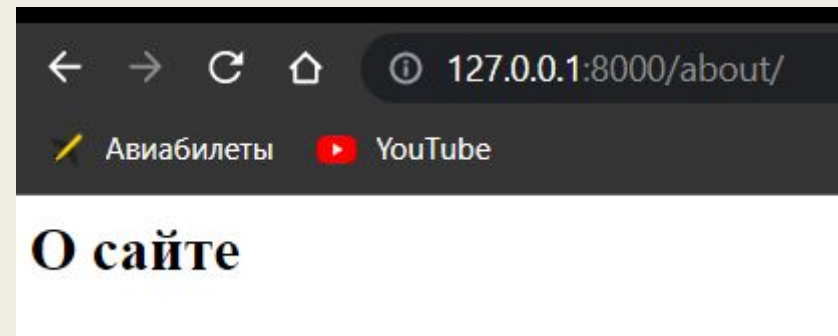
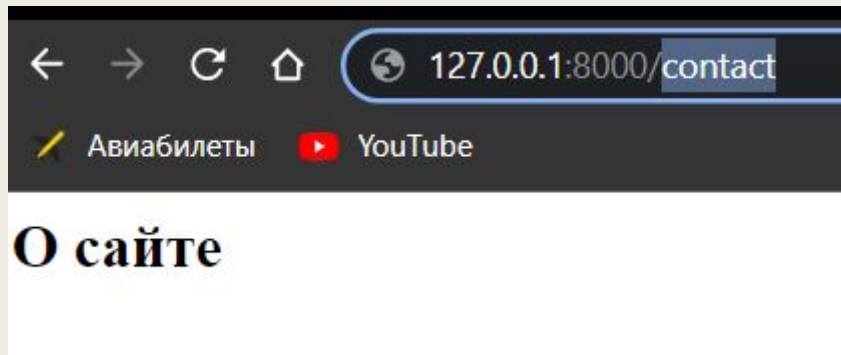
def contact(request):
    return HttpResponseRedirect("/about")

def details(request):
    return HttpResponseRedirectPermanentRedirect("/")
```

Что мы здесь сделали? При обращении к функции `contact` она станет перенаправлять пользователя по пути `"about"`, который будет обрабатываться функцией `about`. А функция `details` станет использовать постоянную переадресацию и перенаправлять пользователя на «корень» (главную страницу) веб-приложения.

```
urlpatterns = [  
    path('', views.index),  
    path('details/', views.details),  
    path('about/', views.about),  
    path('contact/', views.contact),  
    path('products/', views.products),  
    path('products/<int:productid>', views.products),  
    path('users/', views.users),  
    path('users/<int:id>/name/', views.users),  
    path('admin/', admin.site.urls),  
]
```

Теперь в адресной строке браузера наберем адрес страницы contact: `http://127.0.0.1:8000/contact` При этом, поскольку мы задали переадресацию, вместо страницы contact будет загружена страница About. Также в адресной строке браузера наберем адрес страницы details: `http://127.0.0.1:8000/details` При этом, поскольку мы задали переадресацию, вместо страницы details будет загружена главная страница сайта Index.





## Отправка пользователю статусных

### кодов

В пакете `django.http` есть ряд классов, которые позволяют отправлять пользователю определенный статусный код.

Статусный код	Класс
304 (Not Modified)	<code>HttpResponseNotModified</code>
400 (Bad Request)	<code>HttpResponseBadRequest</code>
403 (Forbidden)	<code>HttpResponseForbidden</code>
404 (Not Found)	<code>HttpResponseNotFound</code>
405 (Method Not Allowed)	<code>HttpResponseNotAllowed</code>
410 (Gone)	<code>HttpResponseGone</code>
500 (Internal Server Error)	<code>HttpResponseServerError</code>



кажетс  
я



конец  
лекции