

Лекция 2. **Представления** и **маршрутизация** – ключевые элементы Django

Часть 3



Передача параметров через строку запроса

Есть разные способы передачи параметров, их следует различать:

Параметры могут передаваться **через интернет-адрес (url):**

<http://localhost/index/6/Dirk/>

а могут передаваться через строку запроса:

<http://localhost/index?id=5&name=Dirk/>



Параметры строки запроса

указываются после символа знак вопроса -?

<http://localhost/index/?id=5&name=Dirk/>

Каждый такой параметр – это пара «ключ-значение».

Например: в параметре `id=5` `id` – это ключ, а `5` – это значение.

В параметре `name=Dirk` `name` – это ключ, а `Dirk` – это значение.

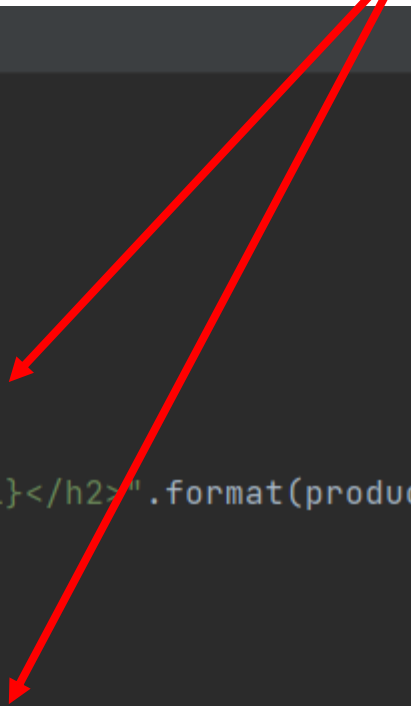
Параметры отделяются один от другого знаком амперсанд - &.

Чтобы получить параметры из строки запроса нужно использовать метод `request.Get.get()`.



Переопределим в файле views.py функции products() и users(), используя метод **request.GET.get()**:

```
views x
14 from django.shortcuts import render
15 from django.http import *
16 from .forms import UserForm
17
18
19 def products(request, productid):
20     category = request.GET.get("cat", "")
21     output = "<h2>Product № {0} Category: {1}</h2>".format(productid, category)
22     return HttpResponse(output)
23
24
25 def users(request):
26     id = request.GET.get("id", 1)
27     name = request.GET.get("name", "Mike")
28     output = "<h2>User</h2><h3>id: {0} name: {1}</h3>".format(id, name)
29     return HttpResponse(output)
30
```



В функцию **products** параметр **productid** будет передаваться через интернет-адрес, а значение параметра **cat** будет извлекаться из строки запроса пользователя :

```
category = request.Get.get("cat", "")
```

"cat" - название параметра строки запроса,

"" - значение по умолчанию на случай ошибки

В функции **users** значения параметров **id** и **name** будут извлекаются из строки запроса пользователя. Значения по умолчанию заданы: **id=1, name="Mike"**.

```
id = request.Get.get("id", 1)
```

```
name = request.Get.get("name", "Mike")
```



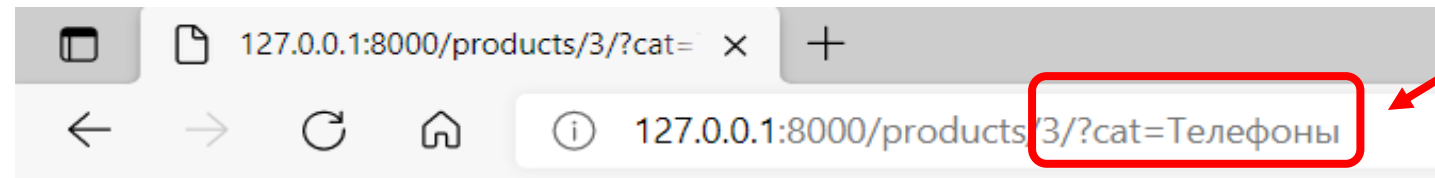
В файле urls.py определим маршруты:

```
urls x
14      2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15      """
16      from django.contrib import admin
17      from django.urls import path, re_path
18      from artist import views
19      from django.views.generic import TemplateView
20
21
22      urlpatterns = [
23          path('', views.index, name='home'),
24          path('products/<int:productid>/', views.products),
25          path('users/', views.users),
26      ]
27
```



запустим отладочный сервер и обратимся к страницам products и users:

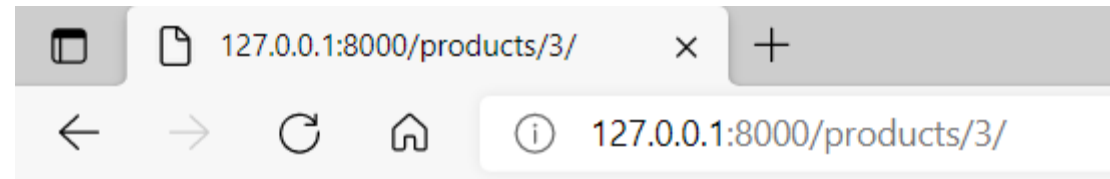
<http://127.0.0.1:8000/products/3/?cat=Телефоны>



Параметр id передается через url-адрес, а параметр cat – через строку запроса

Product № 3 Category: Телефоны

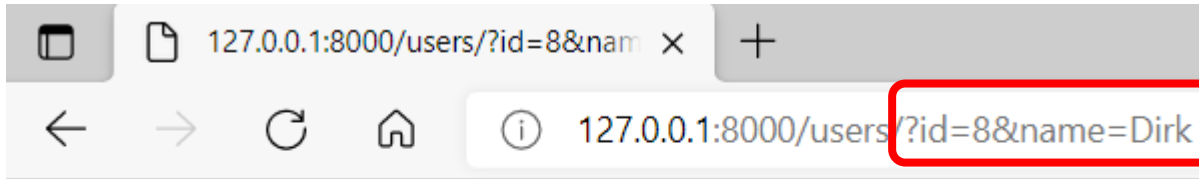
Значение по умолчанию:



Product № 3 Category:



<http://127.0.0.1:8000/users/?id=8&name=Dirk>

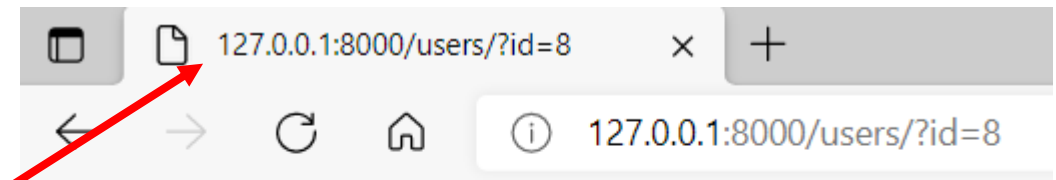


И параметр id и параметр name передаются через строку запроса

User

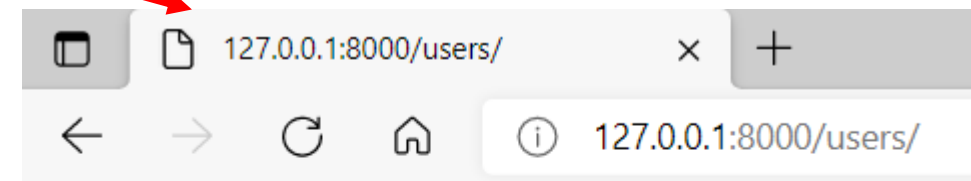
id: 8 name: Dirk

В случае ввода неполных данных будет выведено значение по умолчанию:



User

id: 8 name: Mike



User

id: 1 name: Mike



Обработка исключений при запросах к серверу

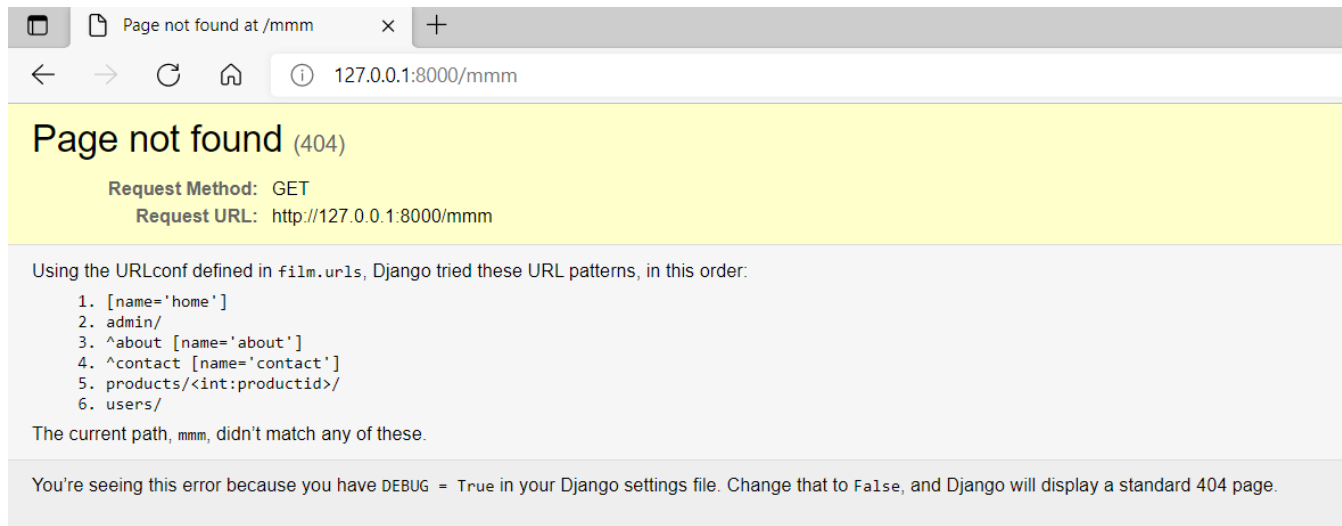
Что мы увидим при переходе на страницу с несуществующим адресом?

Это зависит от того, включен ли режим отладки.

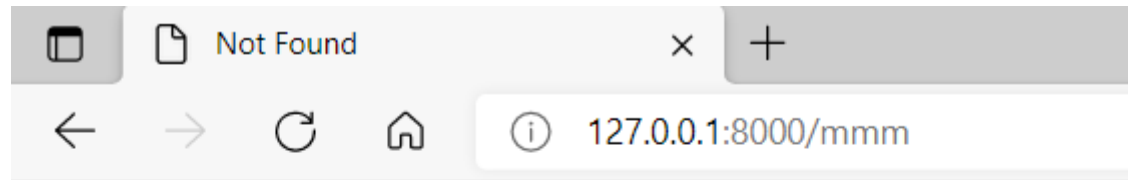
Режим отладки определяется в файле settings.py константой Debug.

Debug = True - режим отладки включен, Debug = False - режим отладки выключен

Если режим отладки включен (Debug = True в файле settings.py), то при переходе на страницу с несуществующим адресом получим **404**:



Если режим отладки выключен (Debug = False и при этом указан хост, например, и локальный сервер: ALLOWED_HOSTS = ['127.0.0.1']), то получим сообщение Not Found



Not Found

The requested resource was not found on this server.



Если мы хотим получать другое сообщение, необходимо:

в файле `views.py` импортировать класс `HttpResponseNotFound` и `Http404`

```
from django.shortcuts import render
```

```
from django.http import HttpResponse
```

```
from django.http import HttpResponseNotFound, Http404
```

создать функцию-обработчик исключения:

```
def pagenotfound(request, exception):
```

```
    return HttpResponseNotFound('<h1>Извините, страница не найдена</h1>')
```

в файле `urls.py` прописать обработчик исключения:

```
handler404 = views.pagenotfound
```

```
from django.shortcuts import render
from django.http import HttpResponse
from django.http import HttpResponseNotFound, Http404
```

```
# Create your views here.
```

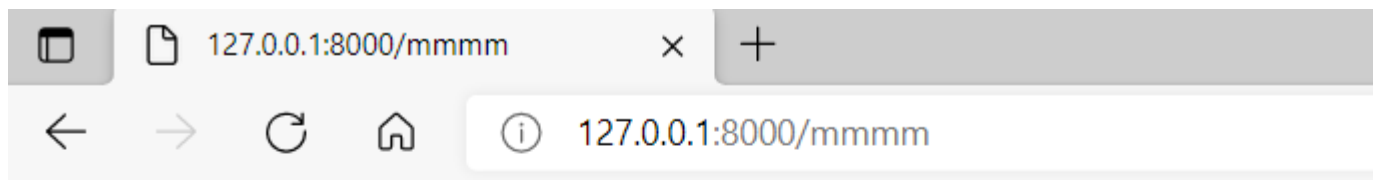
```
def pagenotfound(request, exception):
    return HttpResponseNotFound('<h1>Извините, страница не найдена</h1>')
```

```
from django.contrib import admin
from django.urls import path, re_path
from artist import views
```

```
handler404 = views.pagenotfound
```

```
urlpatterns = [
```

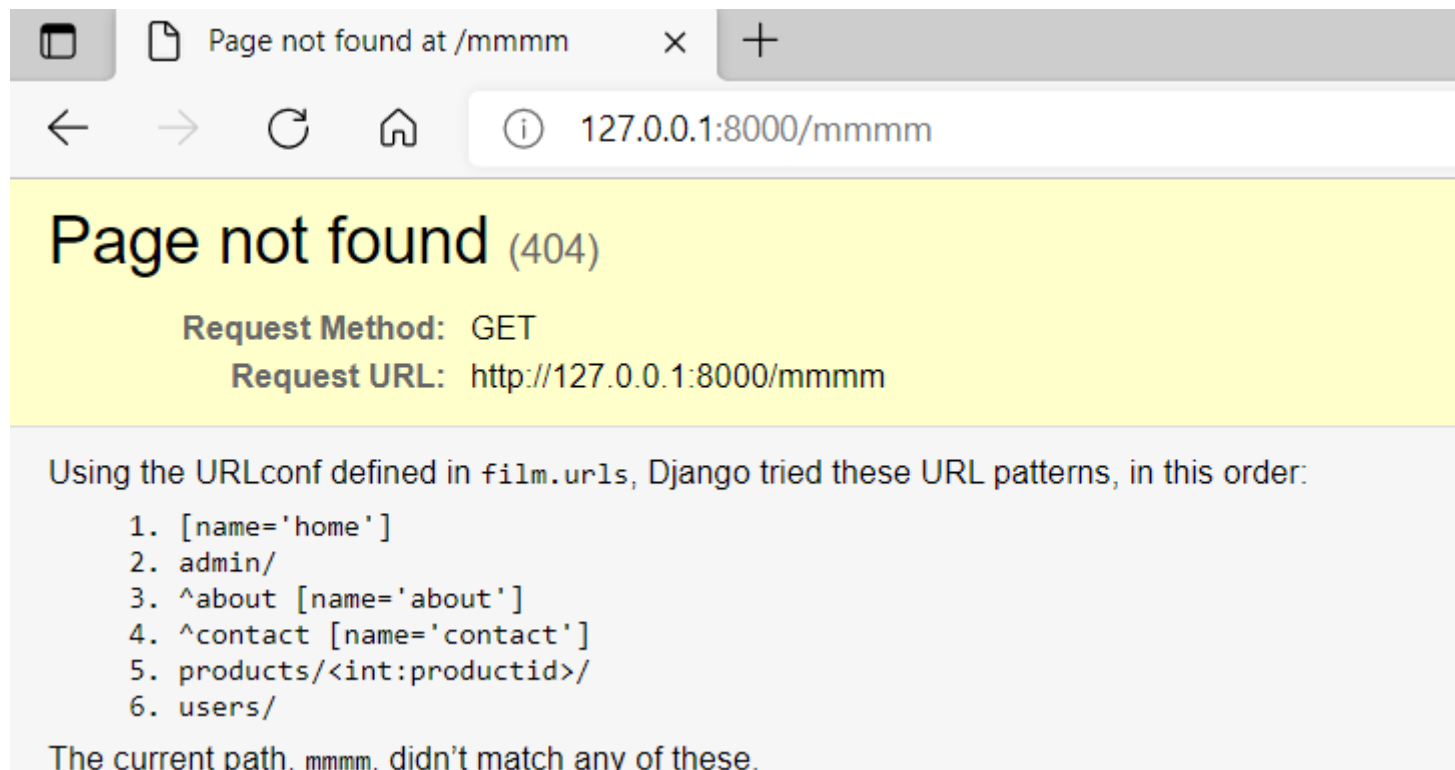




Но это работает строго при условии, что режим отладки выключен:

DEBUG = False

Извините, страница не найдена



Как только мы включим режим отладки: **DEBUG = True**, мы получим такое сообщение



Можно создать такую функцию-обработчик исключения и в ней прописать условие:

```
def proverka(request, year):  
    if int(year) < 2000:  
        raise Http404()  
    output = "<h1> Данные за год {0} </h1>".format(year)  
    return HttpResponseRedirect(output)
```

При этом класс `Http404` также надо импортировать:

```
from django.http import HttpResponseRedirect, Http404
```

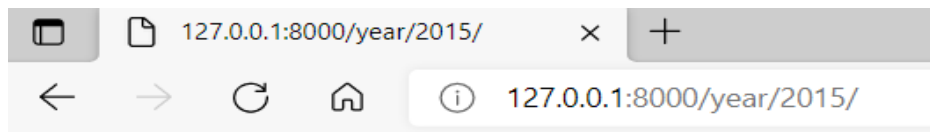
И в файле **`urls.py`** прописать обработчик исключения:

```
handler404 = views.proverka
```



Перейдем на сервер.

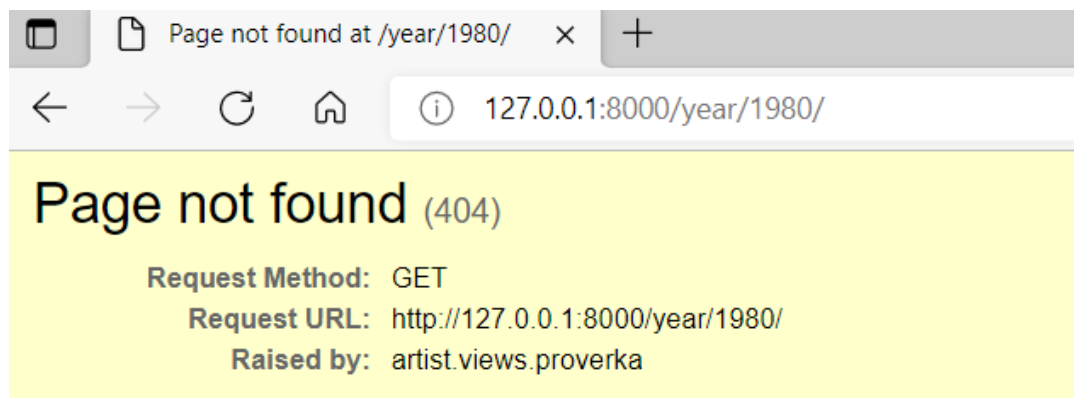
Если год в адресе укажем правильно (≥ 2000), например, 2015, то получим сообщение Данные за год 2015:



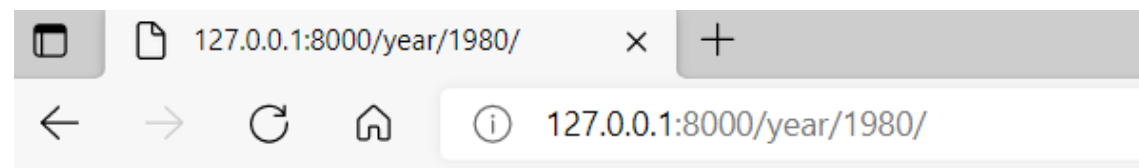
Данные за год 2015

Если год в адресе укажем неправильно (< 2000), например, 1980, то получим сообщение об ошибке.

Если **DEBUG = True** :



если **DEBUG = False**:



Извините, страница не найдена



Аналогично можно определять обработчики для других исключений при запросах к серверу:

- `handler500` – ошибка сервера;
- `handler403` – доступ запрещен;
- `handler400` – невозможно обработать запрос.



Но следует иметь в виду, что все обработчики начинают работать при условии установки:

```
DEBUG = False
```

Более подробно об обработке исключений можно почитать в документации

```
https://djbook.ru/rel3.0/topics/http/views.html
```



Редиректы 301, 302

301 редирект — это способ **постоянного** перенаправления поисковых систем и посетителей сайта на адрес, который отличается от изначально запрашиваемого. Страница перемещена на другой постоянный URL-адрес.

Такой ответ сервера указывает на то, что старый url утратил актуальность, страницу переместили. После переиндексации Яндекс и Google поймут куда вы теперь хотите вести посетителей и станут предлагать пользователям новый адрес.

302 редирект — это переадресация на некоторый временной отрезок, например, на срок от 1-го до 10-ти дней. Страница перемещена временно на другой URL-адрес.



Какая разница между редиректами 302 и 301 ?

Данные типы переадресации разнятся по таким показателям:

- большинство поисковиков отрицательно относятся к 302 редиректу. Так, в некоторых случаях за его использование сайт может попасть под фильтр на 7 дней;
- 301 редирект означает, что документ более не отображается в поиске, тогда как 302 показывает, что отображаются сразу два документа;
- 301 редирект говорит поисковым роботам, что об исходной страничке можно забыть, тогда как 302 – показывает роботам, что стоит продолжать и далее индексировать содержимое страницы;
- 301 редирект передает показатели веб ресурса, ссылочный вес на новый url.



Для создания 301, 302 редиректов используется функция

```
django.shortcuts.redirect
```

Функцию надо импортировать из пакета Django.shortcuts:

```
from django.shortcuts import render, redirect
```



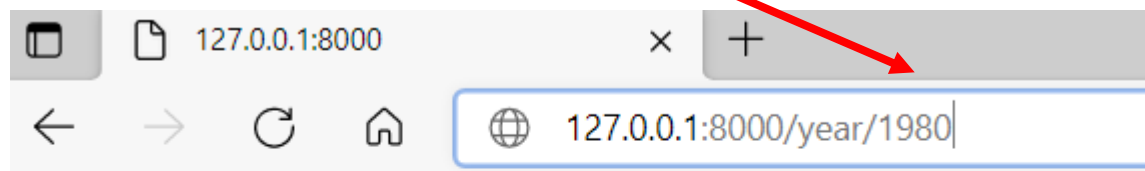
Перепишем ф-ю:

```
def proverka(request, year):  
    if int(year) < 2000:  
        return redirect('home')  
    else:  
        output = "<h1> Данные за год {0} </h1>".format(year)  
        return HttpResponseRedirect(output)
```

Перейдем на сервер и вводим “неправильное” значение года – 1980:



Перейдем на сервер и введем “неправильное” значение года – 1980:

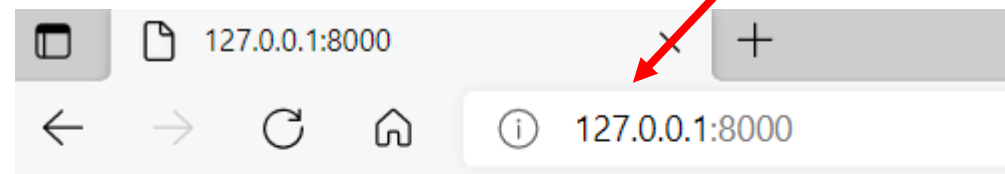


Привет, Артист! Это работает представление `index`

При этом

```
DEBUG = True  
  
ALLOWED_HOSTS = []
```


Происходит переход на страницу ‘home’



Привет, Артист! Это работает представление `index`



В строке терминала увидим код 302.



```
[25/Nov/2021 12:49:42] "GET /year/1980/ HTTP/1.1" 302 0
```

Сообщение говорит нам, что redirect поменялся временно

Чтобы сделать постоянный редирект (301):

```
def proverka(request, year):  
    if int(year) < 2000:  
        return redirect('home', permanent = True)
```



Спасибо за внимание!

