

# Лекция 3.Шаблоны

## Часть 3



## Базовый шаблон.

### Расширение шаблонов HTML-страниц на основе базового шаблона

Если веб-страницы имеют одни и те же структурные элементы, то для того, чтобы сформировать единообразный стиль сайта, надо, чтобы шаблоны имели одинаковую базовую структуру, одни и те же блоки, и при этом для отдельных блоков было определено различное содержимое.

В этом случае можно не определять шаблоны по отдельности, а сформировать и повторно использовать один шаблон, который определит структуру для всех основных блоков. **Такой шаблон называется базовый.**



Например, определим такой базовый шаблон **base.html**:

```
base x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>{% block title %} Заголовок {% endblock title %}</title>
6 </head>
7 <body>
8     <h1>{% block header %}{% endblock header %}</h1>
9     <div>{% block content %}{% endblock content %}</div>
10    <div>Подвал страницы</div>
11 </body>
12 </html>
13
```

Здесь с помощью элементов

**{% block название\_блока %}{% endblock название\_блока %}** определяются отдельные блоки шаблонов.

При этом для каждого блока определяется открывающий элемент:

**{% block название\_блока %}**

и закрывающий элемент:

**{% endblock название\_блока %}**.

Например, блок title имеет структуру: **{% block title %} Заголовок {% endblock title %}**



## Как будет применяться базовый шаблон

Когда другие шаблоны будут применять данный шаблон, то они могут определить для каждого блока какое-то свое содержимое. При этом для каждого блока можно определить содержимое по умолчанию. Так, для блока **title** это строка "Заголовок". И если другие шаблоны, которые будут использовать данный шаблон, не определяют содержимое для блока **title**, то данный блок и в других шаблонах будет использовать строку "Заголовок":

```
<title>{% block title %} Заголовок {% endblock title %}</title>
```

Аналогично определены блоки **header** и **content**:

```
<h1>{% block header %}{% endblock header %}</h1>
```

```
<div>{% block content%}{% endblock content %}</div>
```

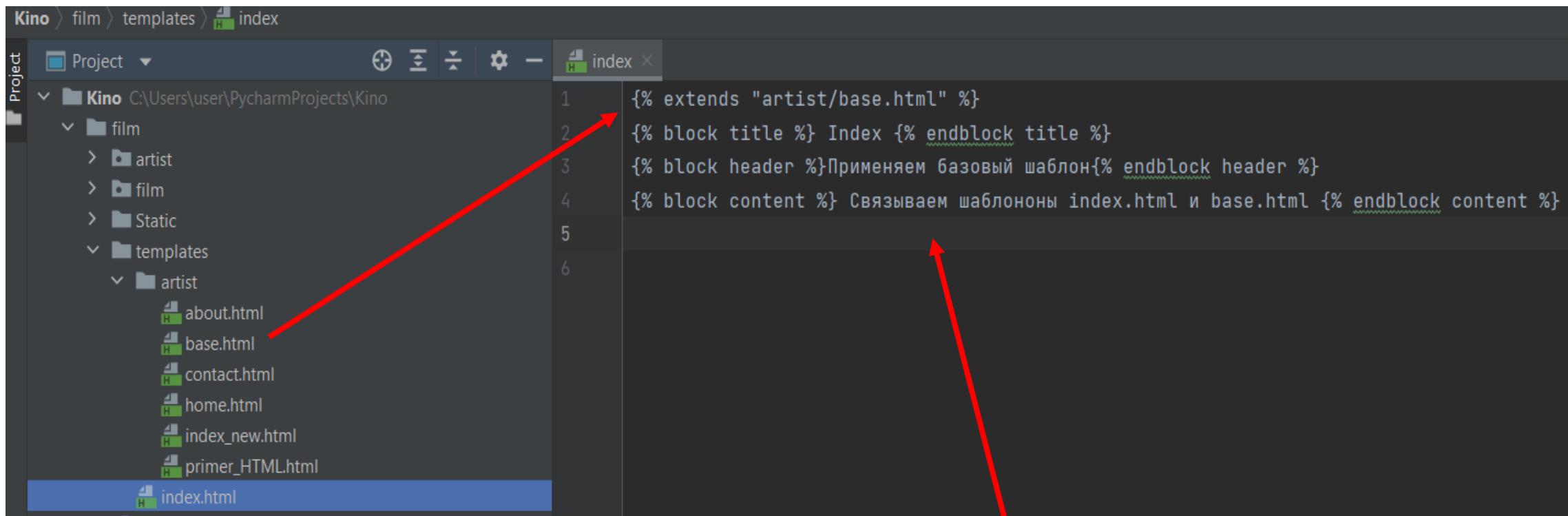
Содержимое по умолчанию для блоков определять не обязательно. Самих блоков при необходимости можно определить сколько угодно.

Кроме того, в базовом шаблоне определен Подвал страницы. И если, допустим, мы хотим сделать его общим для всех страниц, то для него отдельный блок определять не надо.



Применим базовый шаблон **base.html**.

Изменим в каталоге **templates** файл-шаблон главной страницы сайта **index.html**:



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays the file structure of the 'Kino' project. The 'templates' folder is expanded, showing a subfolder 'artist' which contains several HTML files: 'about.html', 'base.html', 'contact.html', 'home.html', 'index\_new.html', 'primer\_HTML.html', and 'index.html'. The 'index.html' file is selected and highlighted in blue. On the right, the editor window shows the content of 'index.html'. The code is as follows:

```
1 {% extends "artist/base.html" %}
2 {% block title %} Index {% endblock title %}
3 {% block header %}Применяем базовый шаблон{% endblock header %}
4 {% block content %} Связываем шаблоны index.html и base.html {% endblock content %}
5
6
```

Two red arrows are present: one points from the 'base.html' file in the project structure to the first line of the code, and the other points from the 'index.html' file in the project structure to the third line of the code.

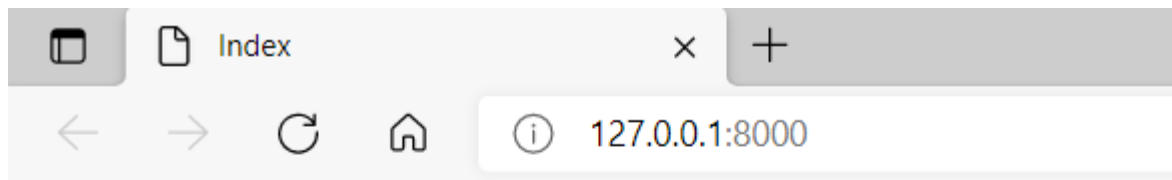
Применим базовый шаблон и затем определим содержимое для блоков **title**, **header** и **content**.

Указывать содержимое для всех блоков базового шаблона необязательно.



Проверим код функции **index()** в представлении **view.py** и перейдем на сервер:

```
8  
9 def index(request):  
10     return render(request, "index.html")
```



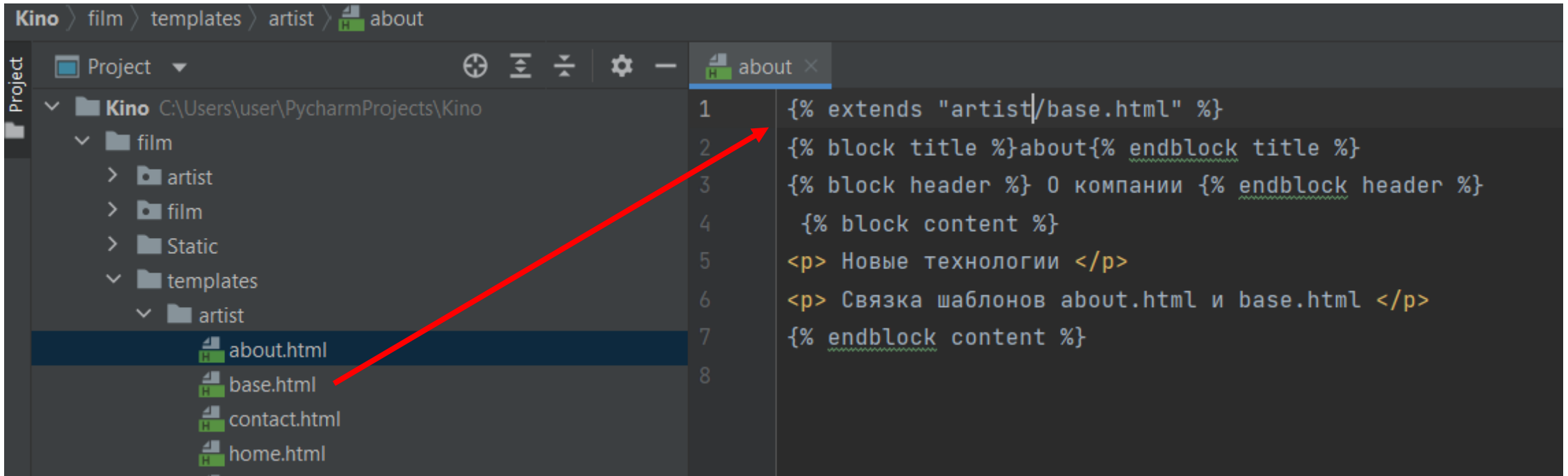
## Применяем базовый шаблон

Связываем шаблоны `index.html` и `base.html`

Подвал страницы



Для формирования другой страницы сайта (**about.html**) также используем базовый шаблон **base.html**:

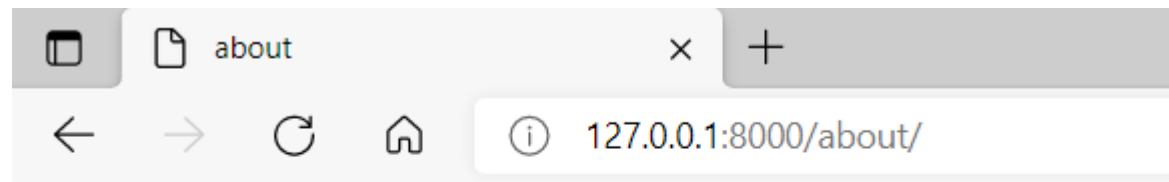


Соответственно уточним функцию **about()** в представлении **view.py**:

```
34
35 def about(request):
36     return render(request, "artist/about.html")
37
```



И перейдем на сервере на страницу **about**:



## **О компании**

Новые технологии

Связка шаблонов about.html и base.html

Подвал страницы





## Выводы

Таким образом, для формирования разных страниц сайта можно использовать один базовый шаблон.

На основе одного базового шаблона можно создавать страницы сайта одинаковой структуры , но с разной информацией.

Если потребуется изменить структуру нескольких веб-страниц сайта, добавить новые элементы или убрать старые, то достаточно будет изменить тот базовый шаблон, который они используют.



## Специальные теги в шаблонах HTML – страниц

Django предоставляет возможность использовать в шаблонах ряд специальных тегов, которые упрощают вывод некоторых данных.

Подробнее: [Встроенные теги и фильтры шаблонов | | документации Django Джанго \(djangoproject.com\)](https://docs.djangoproject.com/ru/2.2/ref/templates/builtins/)

### Даты

Тег `{% now "формат_данных" %}` позволяет вывести системное время. В качестве параметра тегу **now** передается формат данных, который указывает, как форматировать время и дату.



## Даты

Тег `{% now "формат_данных" %}` позволяет вывести системное время. В качестве параметра тегу **now** передается формат данных, который указывает, как форматировать время и дату.

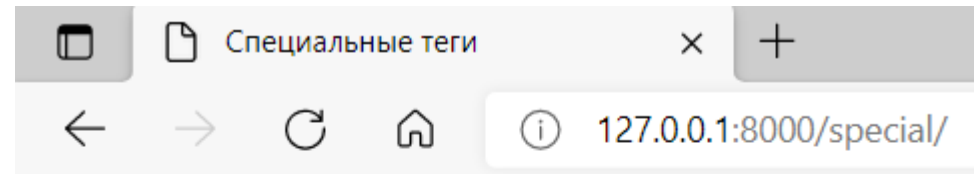
Создадим файл-шаблон **special.html**, в файл **views.py** добавим функцию **special** и пропишем путь.

```
special x
1      <!DOCTYPE html>
2      <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <title>Специальные теги</title>
6      </head>
7      <body>
8          <p>{% now "Y" %}</p>
9          <p>{% now "F j Y" %}</p>
10         <p>{% now "N, j, Y" %}</p>
11         <p>{% now "N j, Y, P" %}</p>
12
13     </body>
14 </html>
15
```

```
38
39 def special(request):
40     return render(request, "artist/special.html")
41
```

```
23 urlpatterns = [
24     path('', views.index, name='home'),
25     path('special/', views.special),
```

Перейдем на сайте по адресу `127.0.0.1:8000/special/`



2021

November 27 2021

Nov., 27, 2021

Nov. 27, 2021, 9:45 a.m.



## if..else

Тег `{% if %} {% endif %}` позволяет выводить в шаблоне определенное содержимое в зависимости от некоторого условия. В качестве параметра тегу `if` передается выражение, которое должно возвращать `True` или `False`.

Например, пусть в представлении передаются в шаблон некоторые значения:

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

```
def index(request):
```

```
    data = {"n" : 5}
```

```
    return render(request, "index.html", context=data)
```



В шаблоне в зависимости от значения переменной  $n$  мы можем выводить определенный контент:

```
{% if n > 0 %}
```

```
<p>Число положительное</p>
```

```
{% endif %}
```

То есть если  $n$  больше 0, то будет выводиться, что число положительное. Если  $n$  меньше или равно 0, ничего не будет выводиться.



С помощью дополнительного тега `{% else %}` можно вывести контент в случае, если условие после if равно False:

```
{% if n > 0 %}
```

```
<p>Число положительное</p>
```

```
{% else %}
```

```
<p>Число отрицательное или равно нулю</p>
```

```
{% endif %}
```



С помощью тега `{% elif %}` можно проверить дополнительные условия, если условие в `if` равно `False`:

```
{% if n > 0 %}
```

```
<p>Число положительное</p>
```

```
{% elif n < 0 %}
```

```
<p>Число отрицательное</p>
```

```
{% else %}
```

```
<p>Число равно нулю</p>
```

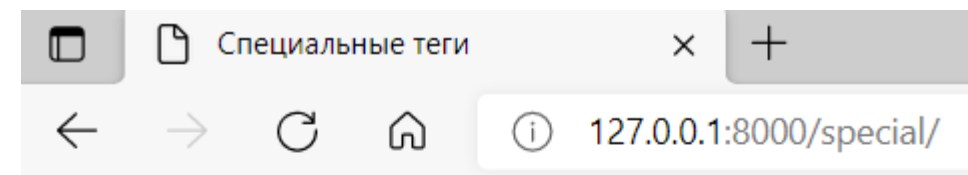
```
{% endif %}
```



Изменим файл **special.html**, функцию **special** в файле **views.py** и перейдем на страницу **special**:

```
special x
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Специальные теги</title>
6 </head>
7 <body>
8   <p>{% now "Y" %}</p>
9   <p>{% now "F j Y" %}</p>
10  <p>{% now "N, j, Y" %}</p>
11  <p>{% now "N j, Y, P" %}</p>
12  {% if city == "Москва" %}
13    <p>Москва - столица нашей Родины</p>
14  {% endif %}
15
16 </body>
17 </html>
```

```
8
9 def special(request):
10   data = {"city": "Москва"}
11   return render(request, "artist/special.html", context=data)
12
```



2021

November 27 2021

Nov., 27, 2021

Nov. 27, 2021, 10:01 a.m.

Москва - столица нашей Родины





## Циклы

Тег **for** позволяет создавать циклы. Этот тег принимает в качестве параметра некоторую коллекцию и пробегается по этой коллекции, обрабатывая каждый ее элемент.

```
{% for element in collection %}
```

```
{% endfor %}
```

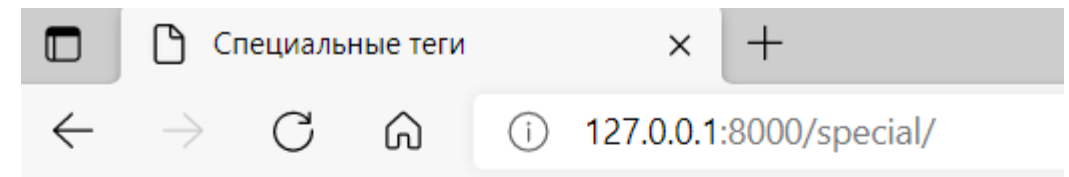


Например, из функции-представления **special** в шаблон **special.html** передается список **cities** :

```
38
39 def special(request):
40     cities = ["Москва", "Санкт-Петербург", "Киев", "Минск", "Новосибирск"]
41     return render(request, "artist/special.html", context={"cities": cities})
```

```
special x
1 <!DOCTYPE html>
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5     <meta charset="UTF-8">
6     <title>Специальные теги</title>
7 </head>
8 <body>
9     <ul>
10         {% for city in cities %}
11         <li>{{ city }}</li>
12         {% endfor %}
13     </ul>
14 </body>
15 </html>
```

обновим страницу:



- Москва
- Санкт-Петербург
- Киев
- Минск
- Новосибирск

Элементы списка **cities** выводятся в шаблоне с помощью тега **for**.

Тег **<ul>** устанавливает маркированный список. Каждый элемент списка должен начинаться с тега **<li>**.



Может случиться, что переданная из представления в шаблон коллекция окажется пустой. На этот случай можно использовать тег **{% empty %}**:

```
<ul>
```

```
    {% for city in cities %}
```

```
        <li>{{ city }}</li>
```

```
    {% empty %}
```

```
        <li>cities array is empty</li>
```

```
    {% endfor %}
```

```
</ul>
```



## Определение переменных

Тег `{% with %}` позволяет определить переменную и использовать ее внутри содержимого тега.

```
{% with name="Иван" age=17 %}
```

```
<div>
```

```
<p>Name: {{ name }}</p>
```

```
<p>Age: {{ age }}</p>
```

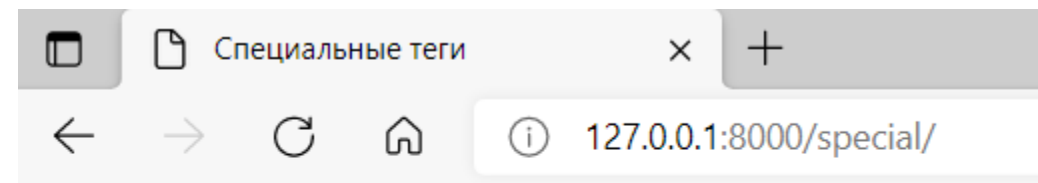
```
</div>
```

```
{% endwith %}
```



Дополним шаблон **special.html** и обновим страницу:

```
special x
6  </head>
7  <body>
8      <ul>
9          {% for city in cities %}
10         <li>{{ city }}</li>
11         {% endfor %}
12     </ul>
13     {% with name="Иван" age=17 %}
14     <div>
15         <p>Name: {{ name }}</p>
16         <p>Age: {{ age }}</p>
17     </div>
18     {% endwith %}
19
20 </body>
21 </html>
```



- Москва
- Санкт-Петербург
- Киев
- Минск
- Новосибирск

Name: Иван

Age: 17



## Формирование URL-адресов в шаблонах

Для того, чтобы сформировать ссылки на страницы в Django используется специальный тег:

`{% url '<URL-адрес или имя маршрута>' [параметры ссылки] %}`

Создадим такой механизм из компонент приложения, работающих в связке:

1. В файле **views.py** создадим функции-представления **menu, index, about, contact**
2. В файле **urls.py** пропишем маршруты к функциям **menu, index, about, contact**
3. В папке **templates/artist** создадим шаблон **new\_menu.html**, в который включим ссылки на маршруты посредством тэгов **url**.



В файле views.py создадим функции-представления:

```
views x
```

```
# получение данных из бд

def menu(request):
    return render(request, "artist/new_menu.html")

def index(request):
    return HttpResponse('Это работает функция index')

def contact(request):
    return HttpResponse('Здесь размещены контакты фирмы Артист')

def about(request):
    return HttpResponse('Здесь размещены сведения о фирме Артист')
```



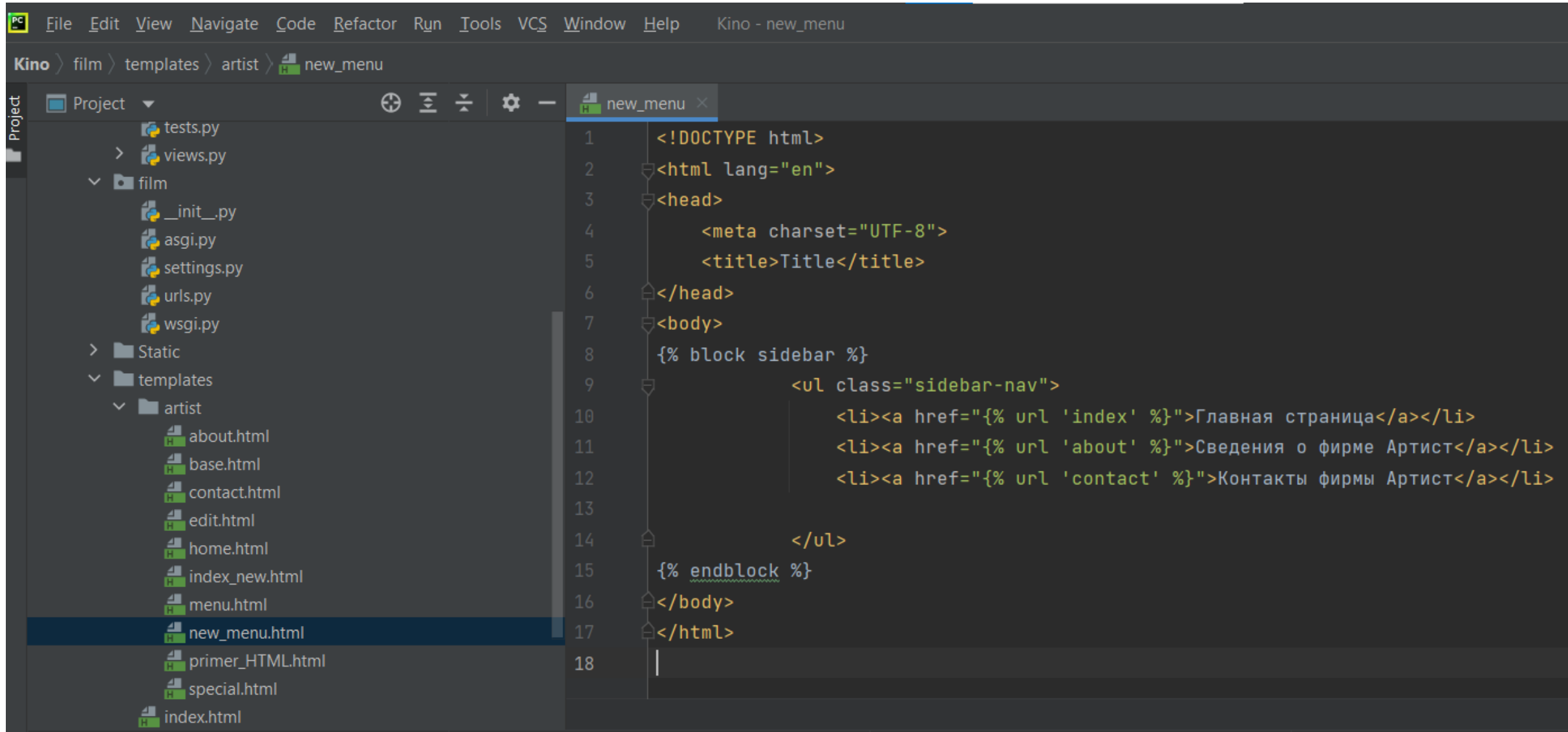
В файле `urls.py` пропишем маршруты, при этом для маршрутов, на которые будет ссылка, укажем имя:

```
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='index'),
24     path('menu/', views.menu),
25     path('about/', views.about, name='about'),
26     path('contact/', views.contact, name='contact'),
27 ]
```





В папке **templates/artist** создадим шаблон **new\_menu.html**:



The screenshot shows the Kino IDE interface. The left sidebar displays the project structure, with the file `new_menu.html` selected under the `templates/artist` directory. The main editor window shows the content of `new_menu.html`, which is a Django template. The template includes a DOCTYPE declaration, HTML and head tags, a meta charset declaration, a title tag, and a body tag. The body contains a Django block named `sidebar` which renders a navigation menu with three links: 'Главная страница', 'Сведения о фирме Артист', and 'Контакты фирмы Артист'.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     {% block sidebar %}
9
10         <ul class="sidebar-nav">
11             <li><a href="{% url 'index' %}">Главная страница</a></li>
12             <li><a href="{% url 'about' %}">Сведения о фирме Артист</a></li>
13             <li><a href="{% url 'contact' %}">Контакты фирмы Артист</a></li>
14         </ul>
15     {% endblock %}
16 </body>
17 </html>
18
```



# Компоненты в связке:

```
views
# получение данных из бд

def menu(request):
    return render(request, "artist/new_menu.html")

def index(request):
    return HttpResponse('Это работает функция index')

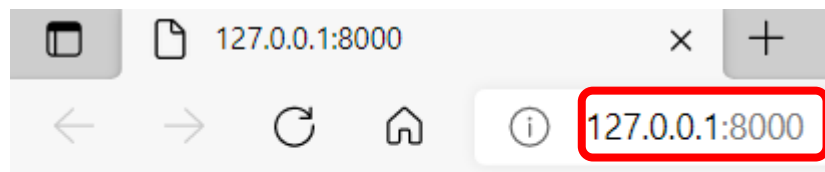
def contact(request):
    return HttpResponse('Здесь размещены контакты фирмы Артист')

def about(request):
    return HttpResponse('Здесь размещены сведения о фирме Артист')
```

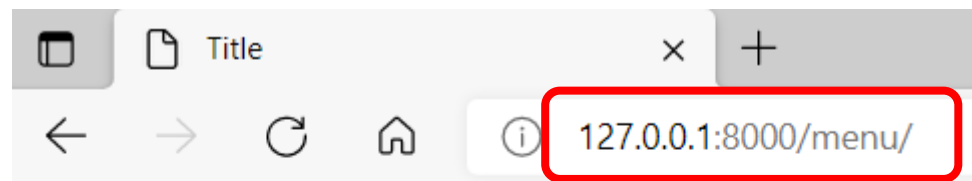
```
new_menu
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8      {% block sidebar %}
9          <ul class="sidebar-nav">
10             <li><a href="{% url 'index' %}">Главная страница</a></li>
11             <li><a href="{% url 'about' %}">Сведения о фирме Артист</a></li>
12             <li><a href="{% url 'contact' %}">Контакты фирмы Артист</a></li>
13          </ul>
14      {% endblock %}
15  </body>
16  </html>
```

```
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='index'),
24     path('menu/', views.menu),
25     path('about/', views.about, name='about'),
26     path('contact/', views.contact, name='contact'),
27 ]
```

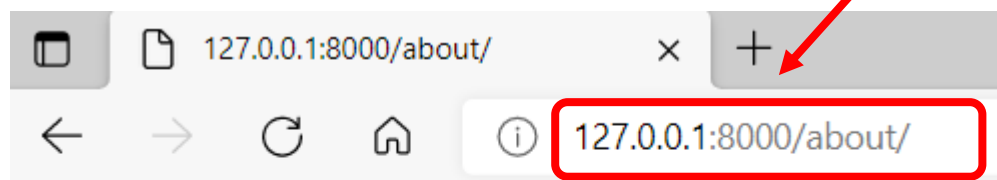




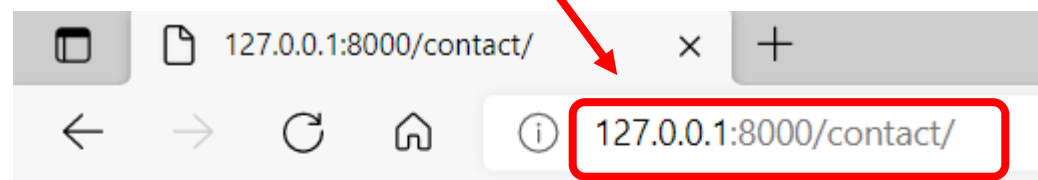
Это работает функция index



- [Главная страница](#)
- [Сведения о фирме Артист](#)
- [Контакты фирмы Артист](#)



Здесь размещены сведения о фирме Артист



Здесь размещены контакты фирмы Артист



Спасибо за внимание.

