

# Лекция 3. Шаблоны

## Часть 1



## Вопросы, которые будут рассмотрены в лекции:

1. Что такое шаблоны
2. HTML - язык разметки гипертекста
3. Класс `TemplateResponse` и функция `render` – в чем разница
4. Передача данных в шаблоны
5. Статичные файлы
6. Каскадные таблицы стилей (Cascading Style Sheets, CSS)
7. Три вида стилей CSS
8. Использование статичных файлов в приложениях Django
9. Класс `TemplateView`
10. Конфигурация шаблонов HTML-страниц
11. Конфигурация шаблонов в файле `settings.py`.
12. Базовый шаблон. Расширение шаблонов HTML-страниц на основе базового шаблона
13. Специальные теги в шаблонах HTML – страниц
14. Формирование URL-адресов в шаблонах



# Часть 1



# Шаблоны (templates)

Шаблоны служат для отображения данных на сайте. С помощью шаблонов оформляется внешний вид приложения.

Шаблоны Django – это HTML-страницы, но тэги Django позволяют вставлять в эти HTML-страницы результаты работы программ на Python.

Обычные браузеры воспринимают только HTML.

HTML расшифровывается как Hyper Text Markup Language, т.е. язык гипертекстовой разметки — основной строительный блок веб-страниц, используется для создания и визуального представления веб-страниц.

HTML был изобретён в 1991 году учёным, Тимом Бёрнсом-Ли (Tim Berners-Lee), и изначально предназначался для обмена документами между учёными различных университетов. Своим изобретением Тим Бёрнс-Ли заложил основы современной сети Internet.



Язык HTML добавляет разметку в обычный текст.

Гипертекст содержит различные ссылки, благодаря которым веб-страницы связываются между собой.

С помощью HTML можно создавать как статические, так и динамические сайты.

HTML является языком, описывающим структуру и семантику содержимого веб-документа.

С помощью тегов, представляющих HTML-элементы, размечается содержимое веб-страницы. Примерами таких элементов являются `<html>`, `<img>`, `<div>` и так далее.

Эти элементы формируют строительные блоки для любого веб-сайта.

[Учебник HTML \(wm-school.ru\)](http://wm-school.ru)



Учебник HTML

→ ↺ 🏠 🔒 https://wm-school.ru/html/default.html

🔍 📄 📖 ⭐ ⚙

wm-school.ru

HTML CSS JavaScript PHP Flex Квиз-тесты

Html начало

HTML уроки:

HTML Введение

HTML Начало

HTML Базовые теги

HTML Элементы

HTML Атрибуты

HTML Форматирование

HTML Мета-теги

HTML Цитаты

HTML Комментарии

HTML Программный код

HTML Ссылки

HTML Стили

HTML Изображения

HTML Цвета

HTML Таблицы

HTML Списки

HTML DIV и SPAN

HTML ID и CLASS

HTML Фреймы

HTML Мультимедиа

Учебник HTML

HTML расшифровывается как **Hyper Text Markup Language**, т.е. язык гипертекстовой разметки — основной строительный блок веб-страниц, используется для создания и визуального представления веб-страниц.

Язык HTML добавляет разметку в обычный текст. Гипертекст содержит различные ссылки благодаря которым веб-страницы связываются между собой. С помощью HTML каждый может создавать как статические так и динамические сайты. HTML является языком, описывающим структуру и семантику содержимого веб-документа. Содержимое веб-страницы размечается с помощью тегов, представляющих HTML-элементы. Примерами таких элементов являются <html>, <img>, <div> и так далее. Эти элементы формируют строительные блоки для любого веб-сайта.

HTML был изобретён в 1991 году учёным, Тимом Бёрнсом-Ли (Tim Berners-Lee), и изначально предназначался для обмена документами между учёными различных университетов. Своим изобретением Тим Бёрнс-Ли заложил основы современной сети Internet.

Существует несколько версий HTML. Стандарт языка непрерывно обновляется и дополняется, следствие этого - почти каждый год выходит новая версия HTML. Версия "HTML 2.0" была первым стандартом HTML спецификации, которая была опубликована в 1995 году. HTML 4.01 является основной версией HTML, которая была опубликована в конце 1999 года, и широко используется в настоящее время. Сегодня наиболее популярной становится версия HTML-5, которая является расширением HTML 4.01, и опубликована в 2012 году.

Все теги HTML

<!-- Комментарий -->

<!DOCTYPE>

<a>

<abbr>

<acronym>

<address>

<applet>

<area>

<article>

<aside>

<audio>

<b>

<base>

Атрибуты HTML-тегов

Список атрибутов

Общие атрибуты

События HTML

Значения атрибутов

MIME-типы



Для хранения шаблонов приложения в корневой папке проекта **film** создадим папку **templates**

В файле **settings.py** в начале файла добавим модуль **os** ( **import os**) и отредактируем переменную **TEMPLATES**:

выше переменной **TEMPLATES** добавим

**TEMPLATE\_DIR = os.path.join(BASE\_DIR, "templates")**

и в переменную внесем **'DIR':[TEMPLATE\_DIR],**

---

Модуль **os** предоставляет множество функций для работы с операционной системой. Подробнее: [Модуль os.path | Python 3 для начинающих и чайников \(pythonworld.ru\)](http://pythonworld.ru)



```
TEMPLATE_DIR = os.path.join(BASE_DIR, "templates")
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [TEMPLATE_DIR],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```





PC File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino - settings

Kino > film > film > settings

Project

- Project
- Kino C:\Users\user\PycharmProjects\Kino
  - film
    - artist
    - film
      - \_\_init\_\_.py
      - asgi.py
      - settings.py
      - urls.py
      - wsgi.py
      - templates
      - db.sqlite3
      - manage.py
    - venv library root
    - main.py
  - External Libraries
  - Scratches and Consoles

```
53  
54 ]  
55  
56 ROOT_URLCONF = 'film.urls'  
57 |  
58 TEMPLATE_DIR = os.path.join(BASE_DIR, "templates")  
59 TEMPLATES = [  
60 {  
61     'BACKEND': 'django.template.backends.django.DjangoTemplates',  
62     'DIRS': [TEMPLATE_DIR],  
63     'APP_DIRS': True,  
64     'OPTIONS': {  
65         'context_processors': [  
66             'django.template.context_processors.debug',  
67             'django.template.context_processors.request',  
68             'django.contrib.auth.context_processors.auth',  
69             'django.contrib.messages.context_processors.messages',  
70         ],  
71     },  
72 },  
73 ]  
74  
75 WSGI_APPLICATION = 'film.wsgi.application'  
76
```



В папке templates создадим файл **index.html** (для этого выполним команду `template/File/HTML File/New HTML File`).

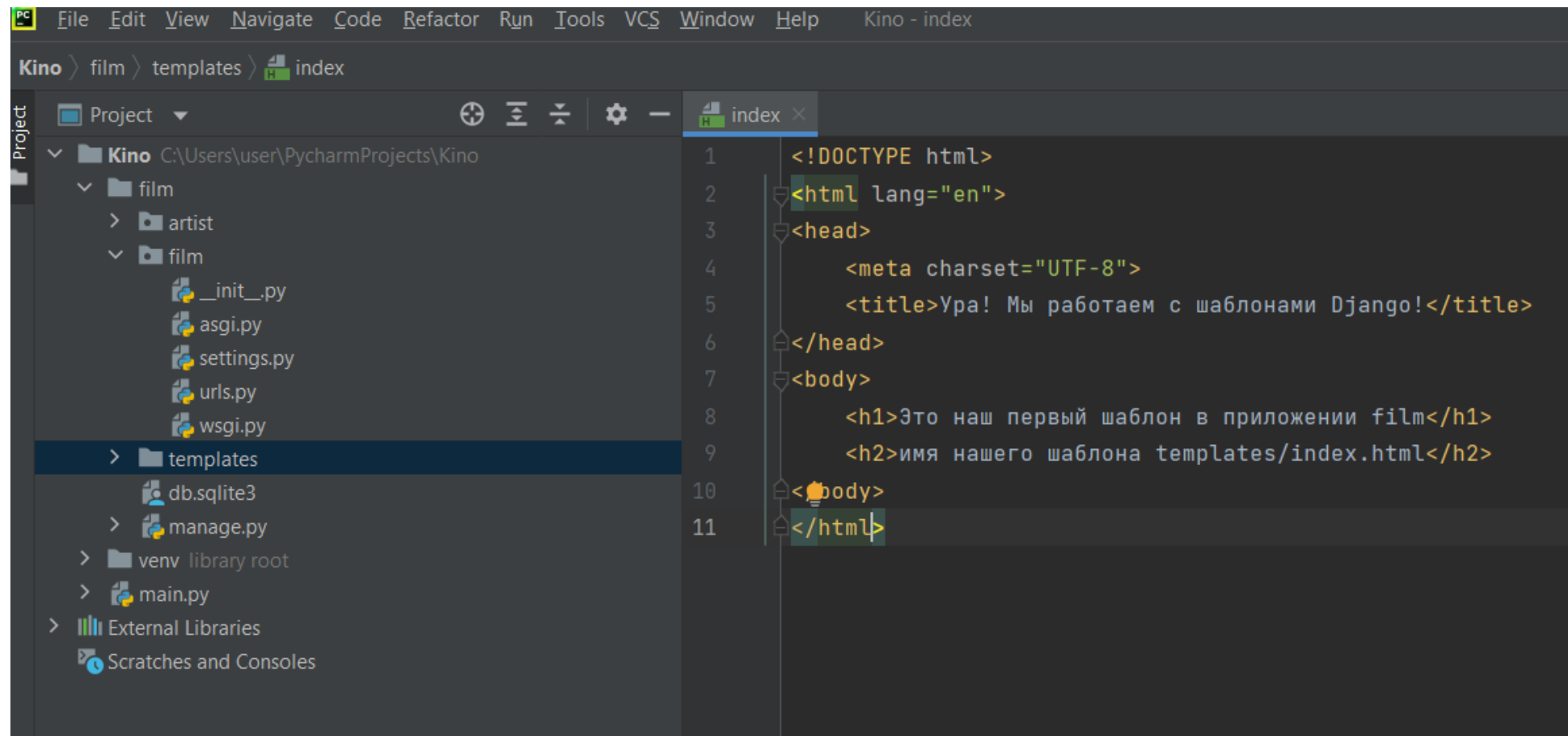
По умолчанию файл **index.html** содержит код:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>

</body>
</html>
```



Изменим файл **index.html** следующим образом :



```
PC File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino - index
Kino > film > templates > index
Project
  Kino C:\Users\user\PycharmProjects\Kino
    film
      artist
      film
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
      templates
      db.sqlite3
      manage.py
      venv library root
      main.py
    External Libraries
    Scratches and Consoles
index
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Ура! Мы работаем с шаблонами Django!</title>
6 </head>
7 <body>
8   <h1>Это наш первый шаблон в приложении film</h1>
9   <h2>имя нашего шаблона templates/index.html</h2>
10 </body>
11 </html>
```



Отредактируем в файле **views.py** приложения **artist** функцию **index** для обработки запроса пользователя:

```
views x
1 from django.http import HttpResponse
2 from django.http import HttpResponseNotFound, Http404
3 from django.shortcuts import render, redirect
4
5 # Create your views here.
6
7
8 def index(request):
9     return render(request, "index.html")
10
```

Функция **index** вызывает функцию **render**, которой передаются объект запроса **request** и имя шаблона в папке **templates** - **"index.html"**.

Функция **render** - выполняет указанный шаблон и возвращает объект **HttpResponse** с полученным содержимым.



```
16 from django.contrib import admin
17 from django.urls import path, re_path
18 from artist import views
19
20 handler404 = views.pagenotfound
21 |
22 urlpatterns = [
23     path('', views.index, name='home'),
```

В файле urls.py проверим маршрут для вызова функции index



Перейдем на сервер : `python manage.py runserver`

В браузере откроется главная страница <http://127.0.0.1:8000>

На странице отобразится:



**Это наш первый шаблон в приложении film**

**имя нашего шаблона templates/index.html**

**Внимание!**

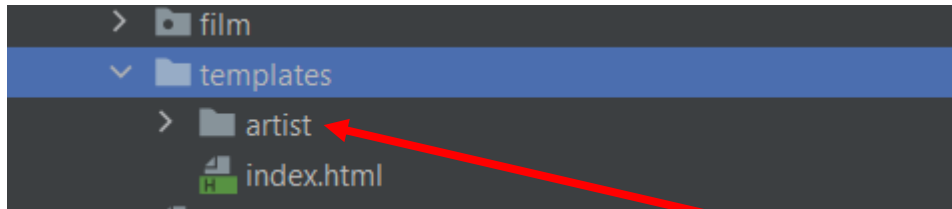
Команду `python manage.py runserver` следует запускать из каталога, в котором находится файл `manage.py`, иначе файл не будет найден и будет выведено сообщение об ошибке.



Каждое из приложений проекта Django может иметь свой набор шаблонов.

Для шаблонов каждого приложения следует создавать отдельную папку.

Для приложения **artist** в папке templates создаем подкаталог **artist**



и в подкаталоге **artist** создаем файл **home.html**



File Edit View Navigate Code Refactor Run Tools VCS Window Help Kino - home

Kino > film > templates > artist > home

Project

- Kino C:\Users\user\PycharmProjects\Kino
  - film
    - artist
      - migrations
      - \_\_init\_\_.py
      - admin.py
      - apps.py
      - models.py
      - primer\_urls.py
      - primer\_view.py
      - proverka.py
      - tests.py
      - views.py
    - film
    - templates
      - artist
        - home.html
        - index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Это файл для приложения artist</title>
6 </head>
7 <body>
8     <h1>Домашняя страница</h1>
9     <h2>шаблон хранится в папке templates/artist, файл home.html</h2>
10 </body>
11 </html>
```






В файле `views.py` приложения **artist** еще раз изменим функцию `index()`:

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

```
    return render(request, "artist/home.html")
```



```
def index(request):  
    return render(request, "artist/home.html")
```

Перейдем на сервер или запустим его `python manage.py runserver`

В браузере по адресу <http://127.0.0.1:8000> откроется новая домашняя страница:





## **Домашняя страница**

**шаблон хранится в папке `templates/artist`, файл `home.html`**

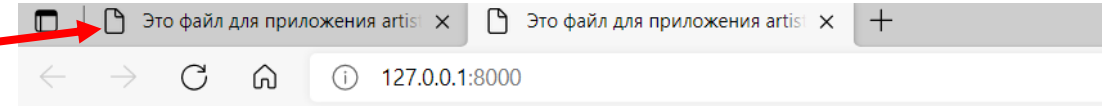


## Класс `TemplateResponse` (шаблонный ответ) - альтернатива функции `render()`

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

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

```
    return render(request, "artist/home.html")
```



Домашняя страница

шаблон хранится в папке `templates/artist`, файл `home.html`

```
from django.template.response import TemplateResponse
```

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

```
    return TemplateResponse(request, "artist/home.html")
```

Наблюдаемый результат один и тот же.



## Класс `TemplateResponse` и функция `render` - в чем разница?

`TemplateResponse` задерживает рендеринг шаблона до тех пор, пока представление не будет завершено. Это позволяет любому промежуточному программному средству шаблона запускать ответ и потенциально изменять шаблон или данные контекста перед тем, как будет создан шаблон. После запуска промежуточного программного обеспечения шаблона шаблон визуализируется, а промежуточное программное обеспечение обычного ответа выполняется на визуализированном контенте до того, как ответ будет возвращен клиенту.

Ярлык `render()` немедленно отображает шаблон и возвращает `HttpResponse`.



## Передача данных в шаблоны

Для вывода самых простых данных в шаблоне используется двойная пара фигурных скобок:

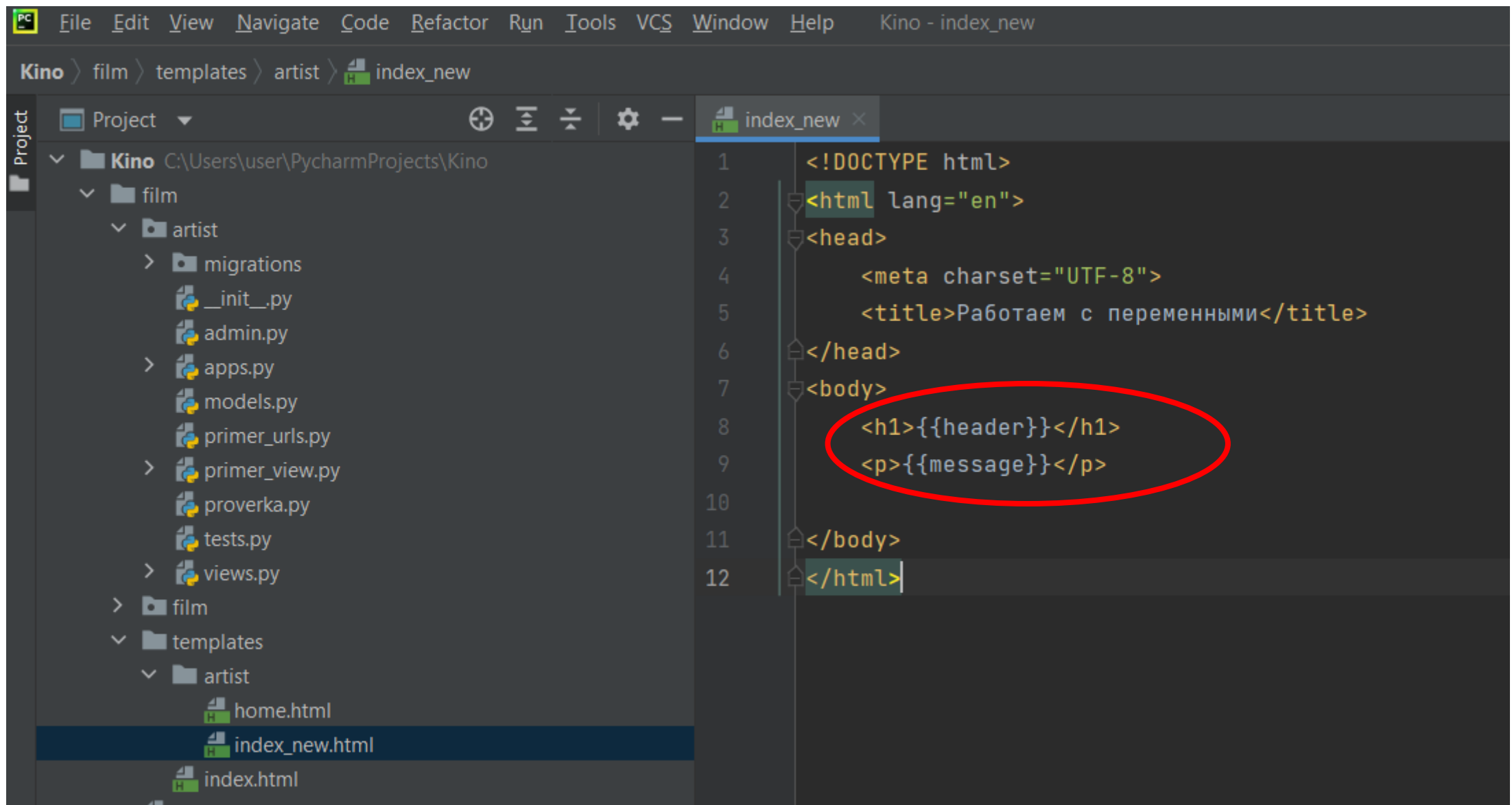
**{{название объекта}}**

В папке `templates/artist` создадим новый шаблон страницы – **`index_new.html`**

Отредактируем код шаблона **`index_new.html`** и введем переменные **`header`** и **`message`**.

Значения в эти переменные будут передаваться из функции-представления файла `view.py`.





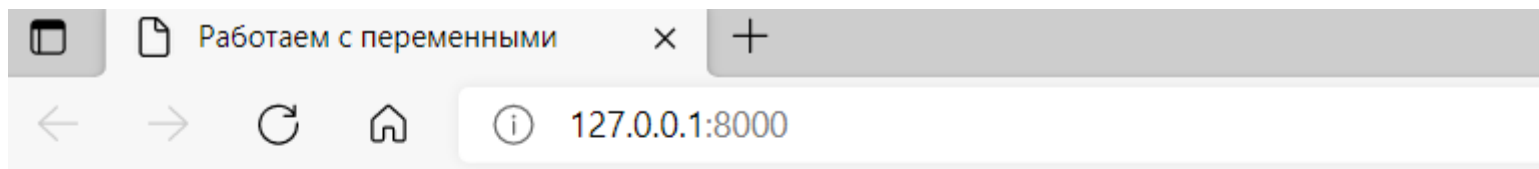
Значения в переменные **header** и **message** будут передаваться из функции-представления файла `view.py`.

В файле `views.py` изменим функцию **index()**:

```
def index(request):  
    data = {"header": "Передача параметров в шаблон Django",  
           "message": "Загружен шаблон templates/artist/index_new.html"}  
    return render(request, "artist/index_new.html", context=data)
```



Перейдем на сервер:



## Передача параметров в шаблон Django

Загружен шаблон templates/artist/index\_new.html

```
def index(request):  
    data = {"header": "Передача параметров в шаблон Django",  
           "message": "Загружен шаблон templates/artist/index_new.html"}  
    return render(request, "artist/index_new.html", context=data)
```





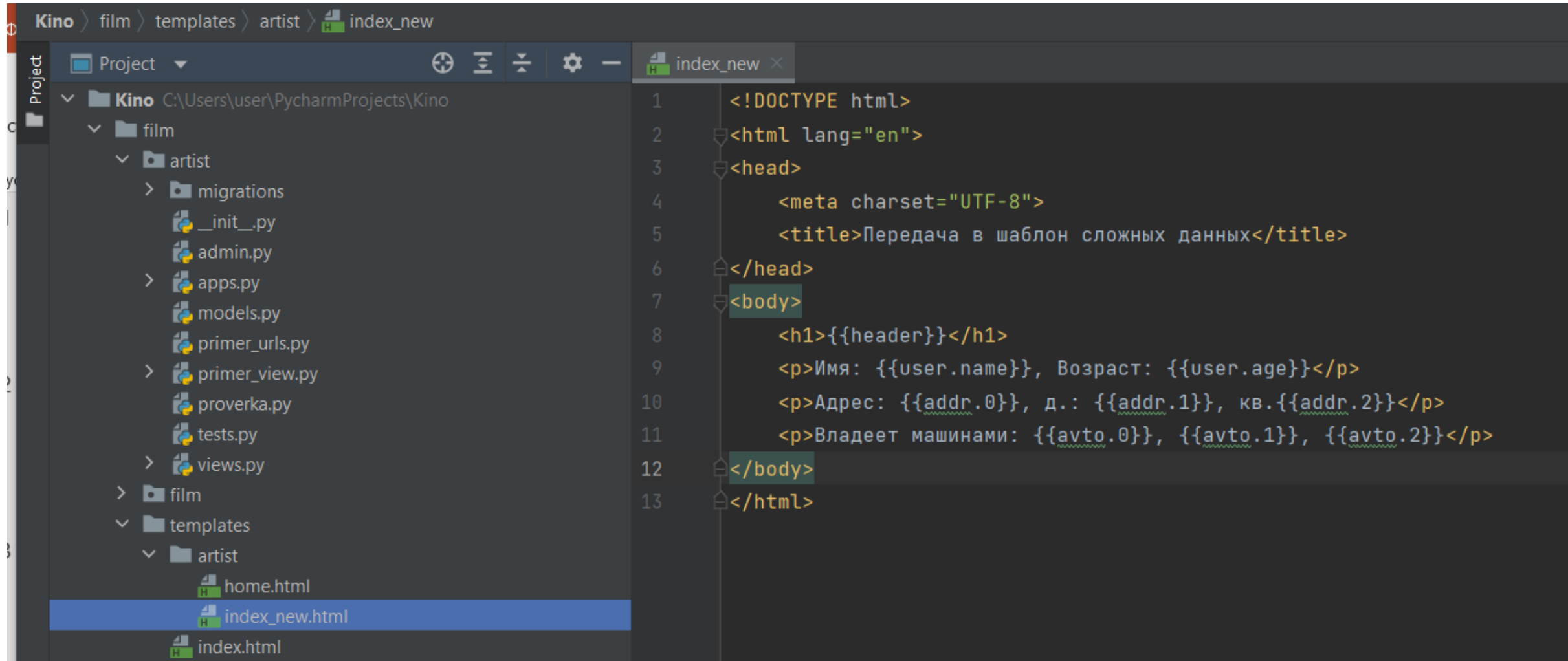
## Передача в шаблон сложных данных

Через шаблон можно передавать пользователю данные более сложные, чем простой текст. Снова изменим функцию **index** в файле представлений (view.py):

```
def index(request):  
    header = "Персональные данные автовладельца" # обычная переменная  
    avto = ["Мазда", "Вольво", "Круз"] # массив  
    user = {"name": "Иван", "age": 25} # словарь  
    addr = ("Русская", 25, 17) # кортеж  
    data = {"header": header, "avto": avto, "user": user, "addr": addr}  
    return render(request, "artist/index_new.html", context=data)
```



Изменим шаблон templates/artist/index\_new.html, чтобы он мог принять новые данные:

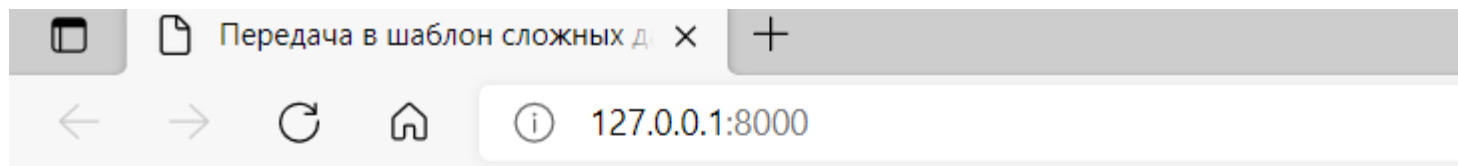


The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays the file structure of the 'Kino' project, with 'templates/artist/index\_new.html' selected. The main editor window shows the content of this HTML template file, which is a Django template. The code includes a DOCTYPE declaration, an HTML lang attribute, a head section with a meta charset and a title, and a body section with an h1 header and three paragraphs of text containing Django template variables. Line numbers 1 through 13 are visible on the left side of the code editor.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Передача в шаблон сложных данных</title>
6 </head>
7 <body>
8     <h1>{{header}}</h1>
9     <p>Имя: {{user.name}}, Возраст: {{user.age}}</p>
10    <p>Адрес: {{addr.0}}, д.: {{addr.1}}, кв.{{addr.2}}</p>
11    <p>Владеет машинами: {{avto.0}}, {{avto.1}}, {{avto.2}}</p>
12 </body>
13 </html>
```



Перейдем на сервер:



## Персональные данные автовладельца

Имя: Иван, Возраст: 25

Адрес: Русская, д.: 25, кв.17

Владеет машинами: Мазда, Вольво, Круз

```
def index(request):  
    header = "Персональные данные автовладельца" # обычная переменная  
    avto = ["Мазда", "Вольво", "Круз"] # массив  
    user = {"name": "Иван", "age": 25} # словарь  
    addr = ("Русская", 25, 17) # кортеж  
    data = {"header": header, "avto": avto, "user": user, "addr": addr}  
    return render(request, "artist/index_new.html", context=data)
```



## Замечание

В случае использования в функции `index` класса `TemplateResponse` в файл `views.py` нужно добавить:

```
from django.template.response import TemplateResponse
```

и в операторе `return` вместо `render` написать `TemplateResponse`



Конец части 1. Продолжение следует..

