

# Лекция 4. Формы

## Часть 1



## Вопросы, которые будут рассмотрены в лекции:

1. Что такое форма
2. Где могут быть размещены формы
3. Как объект формы передается в шаблон
4. Как связаны роль формы и запросы POST и GET
5. Виджеты Django
6. Типы полей формы
7. Настройка вида формы и полей формы
8. Валидация данных. Правила валидации.
9. Детальная настройка полей формы
10. Применение собственных стилей к полям форм



# Часть 1



## Формы.

Форма – это объект, включающий набор полей (виджетов) на веб-странице для получения данных от пользователя для последующей передачи их на сервер.

Виджет – представление элемента на HTML-странице. Отвечает за внешний вид поля.

Каждая форма определяется в виде отдельного класса, который расширяет класс **forms.Form**.

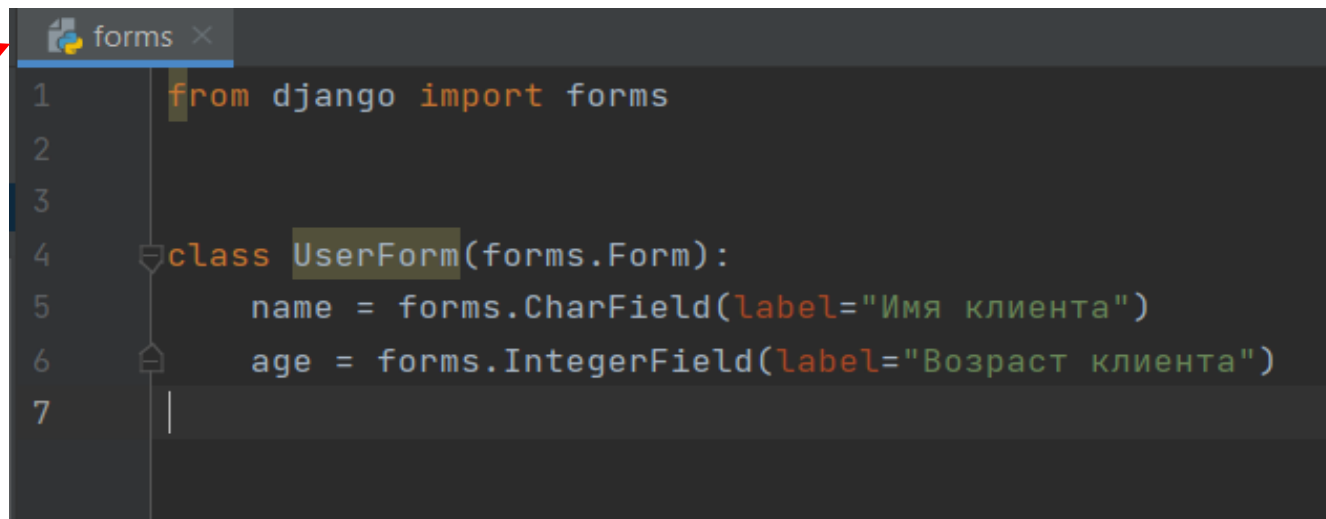
Классы размещаются внутри проекта, где они используются.

Формы могут быть помещены в отдельный файл, например, **forms.py**.

Также формы могут размещаться внутри уже имеющихся в приложении файлов, например, в **views.py** или **models.py**.



Создадим в приложении **artist**  
проекта **film** новый файл **forms.py**:



```
1  from django import forms
2
3
4  class UserForm(forms.Form):
5      name = forms.CharField(label="Имя клиента")
6      age = forms.IntegerField(label="Возраст клиента")
7
```

В файле **views.py**  
изменим представление **index**:



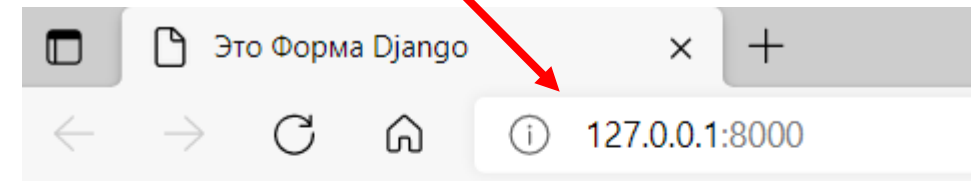
```
4  from django.shortcuts import render, redirect
5  from .forms import UserForm
6  from django.template.response import TemplateResponse
7
8  # Create your views here.
9
10 def index(request):
11     userform = UserForm()
12     return render(request, "index.html", {"form": userform})
13
```



Отредактируем шаблон index.html и перейдем на сайт:

```
index x
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title> Это Форма Django </title>
6  </head>
7  <body>
8      <table>
9          {{ form }}
10     </table>
11 </body>
12 </html>
```

```
22
23 urlpatterns = [
24     path('', views.index, name='home'),
```



Имя клиента:

Возраст клиента:

Объект формы передается в шаблон в виде переменной **form**.



## Использование в формах POST-запросов

Создадим форму, в которой можно не только вводить данные, но и отправлять их на сервер.

Сначала изменим шаблон index.html:

```
index x
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title> Это Форма Django </title>
6  </head>
7  <body>
8      <form method="POST">
9          {% csrf_token %}
10         <table>
11             {{ form }}
12         </table>
13         <input type="submit" value="Отправить" >
14     </form>
15 </body>
16 </html>
```

Теперь Django `{% csrf_token %}` позволяет защитить приложение от CSRF (Cross-Site Request Forgery) атак, добавляя в форму в виде скрытого поля csrf-токен.



Далее изменим функцию `index()` в файле `views.py`:

```
views x
11 from django.http import *
12 from .forms import UserForm
13
14
15 def index(request):
16     if request.method == "POST":
17         # получение значения поля Имя
18         name = request.POST.get("name")
19         # получение значения поля Возраст
20         age = request.POST.get("age")
21         output = "<h2>Пользователь</h2><h3>Имя - {0}, Возраст - {1}</h3>".format(name, age)
22         return HttpResponse(output)
23     else:
24         userform = UserForm()
25         return render(request, "index.html", {"form": userform})
26
```

Представление обрабатывает сразу два типа запросов: **GET** и **POST**.

Для определения типа запроса делается проверка значения `request.method`.

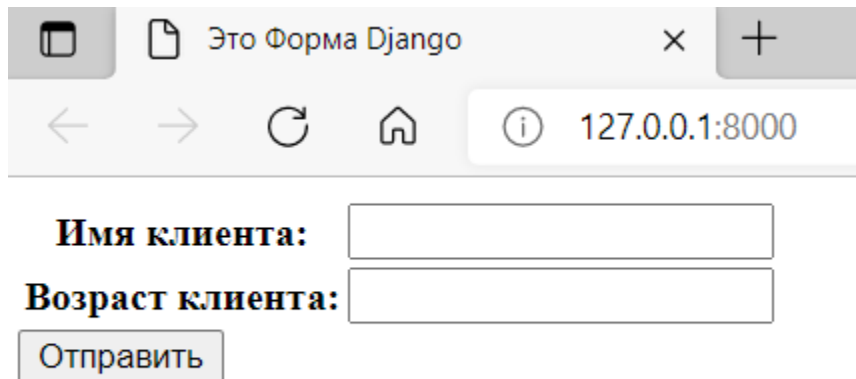
Если запрос типа **POST**, то будет форма для отправки данных.

Если запрос представляет тип **GET** (ветка `else`), то форма будет для ввода данных.



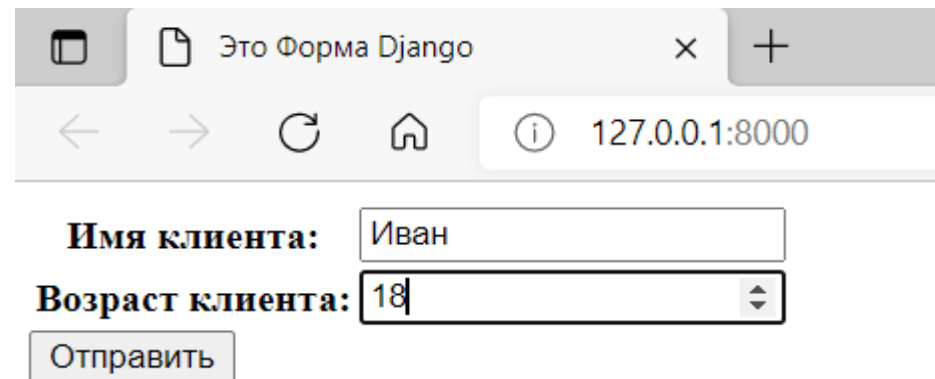


Переходим на сервер:



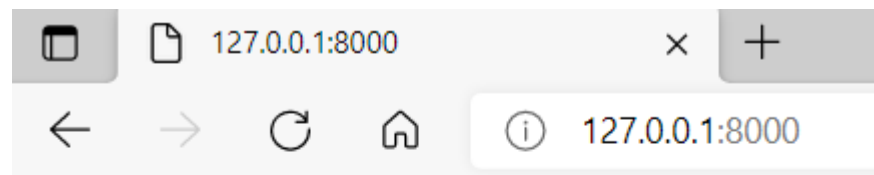
The screenshot shows a web browser window with the title "Это Форма Django". The address bar displays "127.0.0.1:8000". The form contains two input fields: "Имя клиента:" and "Возраст клиента:". Below the fields is a button labeled "Отправить".

Если запрос будет типа POST (`request.method == "POST"`)), то данные будут отправлены по нажатию кнопки Отправить. Отправляемые данные будут присвоены переменным `name` и `age`, а их значения переменной `output`. Затем значения из `output` будут отправлены через объект `HttpResponse`. В данном примере ответ отправляется пользователю на ту же HTML-страницу.



The screenshot shows the same web browser window, but the "Имя клиента:" field now contains the text "Иван" and the "Возраст клиента:" field contains the number "18". The "Отправить" button remains visible.

Введем в поля формы значения Иван и 18 и нажмем Отправить. Получим:



The screenshot shows the same web browser window, but the address bar now displays "127.0.0.1:8000". The form fields and the "Отправить" button are not visible in this screenshot.

**Пользователь**

**Имя - Иван, Возраст – 18**



## Виджеты Django

Поля формы при генерации разметки используют определенные виджеты из пакета **forms.widgets**. Например, класс **CharField** использует виджет **forms.widgets.TextInput**, а **ChoiceField** использует **forms.widgets.Select**.

Но есть ряд виджетов, которые по умолчанию не используются полями форм, но мы их можем применять:

**PasswordInput:** генерирует поле для ввода пароля `<input type="password" >`

**HiddenInput:** генерирует скрытое поле `<input type="hidden" >`

**MultipleHiddenInput:** генерирует набор скрытых полей



**TextArea:** генерирует многострочное текстовое поле `<textarea></textarea>`

**RadioSelect:** генерирует список переключателей (радиокнопок) `<input type="radio" >`

**CheckboxSelectMultiple:** генерирует список флажков `<input type="checkbox" >`

**TimeInput:** генерирует поле для ввода времени (например, 12:41 или 12:41:32)

**SelectDateWidget:** генерирует три поля select для выбора дня, месяца и года

**SplitHiddenDateTimeWidget:** использует скрытое поле для хранения даты и времени

**FileInput:** генерирует поле для выбора файла



## Типы полей формы

В формах Django доступны следующие классы для создания полей форм:

**forms.BooleanField:** создает поле `<input type="checkbox" >` для выбора решения. Возвращает значение Boolean: True - если флажок отмечен и False - если флажок не отмечен. По умолчанию устанавливается виджет **CheckboxInput**.

### Пример

```
from django import forms
```

```
class UserForm(forms.Form)
```

```
    basket= forms.BooleanField(label="Положить товар в корзину")
```



**forms.NullBooleanField:** создает следующую разметку:

```
<select>
```

```
<option value="1" selected="selected">Неизвестно</option>
```

```
<option value="2">Да</option>
```

```
<option value="3">Нет</option>
```

```
</select>
```

По умолчанию использует виджет **NullBooleanSelect**.



**forms.CharField:** предназначен для ввода текста и создает следующую разметку:

```
<input type="text">
```

По умолчанию использует виджет **TextInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
name= forms.CharField(label="Имя клиента")
```



**forms.EmailField:** предназначен для ввода адреса электронной почты и создает следующую разметку:

```
<input type="email">
```

По умолчанию использует виджет **EmailInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
email= forms.EmailField(label="электронный адрес", help_text="обязательный символ – @")
```



**forms.GenericIPAddressField:** предназначен для ввода IP-адреса в формате IP4v или IP6v и создает следующую разметку:

```
<input type="text">
```

По умолчанию использует виджет **TextInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
ip_adres= forms.GenericIPAddressField (label="IP адрес", help_text="Пример формата 192.0.2.0")
```





**forms.RegexField (regex="регулярное\_выражение")**: предназначен для ввода текста, который должен соответствовать определенному регулярному выражению. Создает текстовое поле:

**<input type="text">**

По умолчанию использует виджет **TextInput** и метод **RegexValidator**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
reg_text= forms.RegexField(label="Текст", regex="^[0-9][A-F][0-9]$")
```



**forms.SlugField()**: предназначен для ввода текста, который условно называется "**slug**", то есть последовательность символов в нижнем регистре, чисел, дефисов и знаков подчеркивания. Создает текстовое поле:

**<input type="text">**

По умолчанию использует виджет **TextInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
slug_text= forms.SlugField(label="Введите текст")
```



**forms.URLField()**: предназначен для ввода универсального указателя ресурса(ссылок). Создает следующее поле:

**<input type="url">**

По умолчанию использует виджет **URLInput**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
url_text= forms.URLField(label="Введите URL", help_text="Например, http://www.google.com")
```



**forms.UUIDField()**: предназначен для ввода UUID (универсального уникального идентификатора). Создает следующее поле:

**<input type="text">**

По умолчанию использует виджет **TextInput**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
uuid_text= forms.UUIDField(label="Введите UUID", help_text="Формат  
xxxxxxxx_xxxx_xxxx_xxxx_xxxxxxxxxxxxxx")
```



**forms.ComboField(fields=[field1, field2,...]):** аналогичен обычному текстовому полю за тем исключением, что требует, чтобы вводимый текст соответствовал требованиям тех полей, которые передаются через параметр `fields`. Создает следующее поле:

**<input type="text">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    combo_text= forms.ComboField(label="Введите URL",fields=[ forms.URLField(),  
    forms.CharField(max_length=20)])
```



**forms.FilePathField(path="каталог файлов"):** создает список select, который содержит все папки и файлы в определенном каталоге:

```
<select>
```

```
<option value="folder/file1">folder/file1</option>
```

```
<option value="folder/file2">folder/file2</option>
```

```
<option value="folder/file3">folder/file3</option>
```

```
//.....
```

```
</select>
```

По умолчанию использует виджет **Select**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    file_path= forms.FilePathField(label="Выберите файл:", path="C:/Python/")
```



**forms.FileField()**: предназначен для выбора файла. Создает следующее поле:

**<input type="file">**

По умолчанию использует виджет **ClearableFileInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    file= forms.FileField(label="Файл")
```



**forms.ImageField()**: предназначен также для выбора файла изображения.

Создает следующее поле:

**<input type="file">**

По умолчанию использует виджет **ClearableFileInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    file= forms.ImageField (label="Изображение")
```





**forms.DateField():** предназначено для ввода даты. В создаваемое поле вводится текст, который может быть сконвертирован в дату, например, 2017-12-25 или 11/25/17.

Создает следующее поле:

**<input type="text">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    date= forms.DateField(label="Введите дату:")
```



**forms.TimeField()**: предназначен ввода времени, например, 14:30:59 или 14:30.

Создает следующее поле:

**<input type="text">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    time= forms.TimeField (label="Введите время:")
```



**forms.DateTimeField()**: предназначен ввода даты и времени, например, 2017-12-25 14:30:59 или 11/25/17 14:30.

Создает следующее поле:

**<input type="text">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    date_time= forms.DateTimeField (label="Введите дату и время:")
```



**forms.DurationField()**: предназначен для ввода промежутка времени. Вводимый текст должен соответствовать формату "DD HH:MM:SS", например, 2 1:10:20 (2 дня 1 час 10 минут 20 секунд). Создает следующее поле:

**<input type="text">**

По умолчанию использует виджет **TextInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    time_delta= forms.DurationField (label="Введите промежуток времени")
```



**forms.SplitDateTimeField()**: создает два текстовых поля для ввода соответственно даты и времени:

```
<input type="text" name="_0" >
```

```
<input type="text" name="_1" >
```

По умолчанию использует виджет **SplitDateTimeWidget**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    time_delta= forms.SplitDateTimeField (label="Введите дату и время")
```



**forms.IntegerField()**: предназначен для ввода целых чисел.

Создает следующее поле:

**<input type="number">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    num= forms.IntegerField("label="Введите целое число")
```



**forms.DecimalField():** предназначен для ввода чисел.

Создает следующее поле:

**<input type="number">**

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    num= forms.DecimalField ("label="Введите десятичное число", decimal_places=2)
```



**forms.FloatField()**: предназначен для ввода чисел.

Создает следующее поле:

**<input type="number">**

По умолчанию использует виджет **NumberInput**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    num= forms.FloatField ("label="Введите число с плавающей точкой")
```





**forms.ChoiceField(choices=кортеж\_кортежей):** выбор данных из списка, генерирует список **select**, каждый из его элементов формируется на основе отдельного кортежа. Например, следующее поле:

**forms.ChoiceField(choices=((1, "English"), (2, "German"), (3, "French")))**

будет генерировать следующую разметку:

```
<select>
```

```
<option value="1">English</option>
```

```
<option value="2">German</option>
```

```
<option value="3">French</option>
```

```
</select>
```

По умолчанию использует виджет **Select**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    country= forms.ChoiceField (label="Выберите  
    страну",  
    choices=((1,Россия),(2,Франция),(3,Италия))
```



**forms.TypeChoiceField(choises=кортеж\_кортежей, coerce=функция\_преобразования, empty\_value=None):** также генерирует список select на основе кортежа. Однако дополнительно принимает функцию преобразования, которая преобразует каждый элемент. И также принимает параметр empty\_value, который указывает на значение по умолчанию.

Создает на HTML -странице разметку:

```
<select>

    <option value="1">Date 1</option>

    <option value="2">Date 2</option>

    <option value="3">Date 3</option>

</select>
```

По умолчанию использует виджет **Select**.

Пример:

```
from Django import forms

class UserForm(form.Form)

    city= forms.TypeChoiceField (label="Выберите
    город",
    empty_value=None,
    choices=((1,Москва),(2, Париж),(3,Милан))
```



**forms.MultipleChoiceField(choices=кортеж\_кортежей):** также генерирует список **select** на основе кортежа, как и **forms.ChoiceField**, добавляя к создаваемому полю атрибут **multiple="multiple"**. То есть список поддерживает множественный выбор.

По умолчанию использует виджет **SelectMultiple**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    countries= forms.MultipleChoiceField (label="Выберите страны",  
    choices=((1,Россия),(2,Франция),(3,Италия),(4,Испания))
```



**forms.TypedMultipleChoiceField(choises=кортеж\_кортежей, coerce=функция\_преобразования, empty\_value=None):** аналог forms.TypeChoiceField для списка с множественным выбором.

По умолчанию использует виджет **SelectMultiple**.

Пример:

```
from Django import forms
```

```
class UserForm(form.Form)
```

```
    countries= forms.MultipleChoiceField (label="Выберите страны",
```

```
    empty_value=None,
```

```
    choices=((1,Россия),(2,Франция),(3,Италия),(4,Испания))
```



Конец части 1. Продолжение следует..

