Запросы в Django
httpRequest. GET и POST

Формы HTML

<input type='text'>

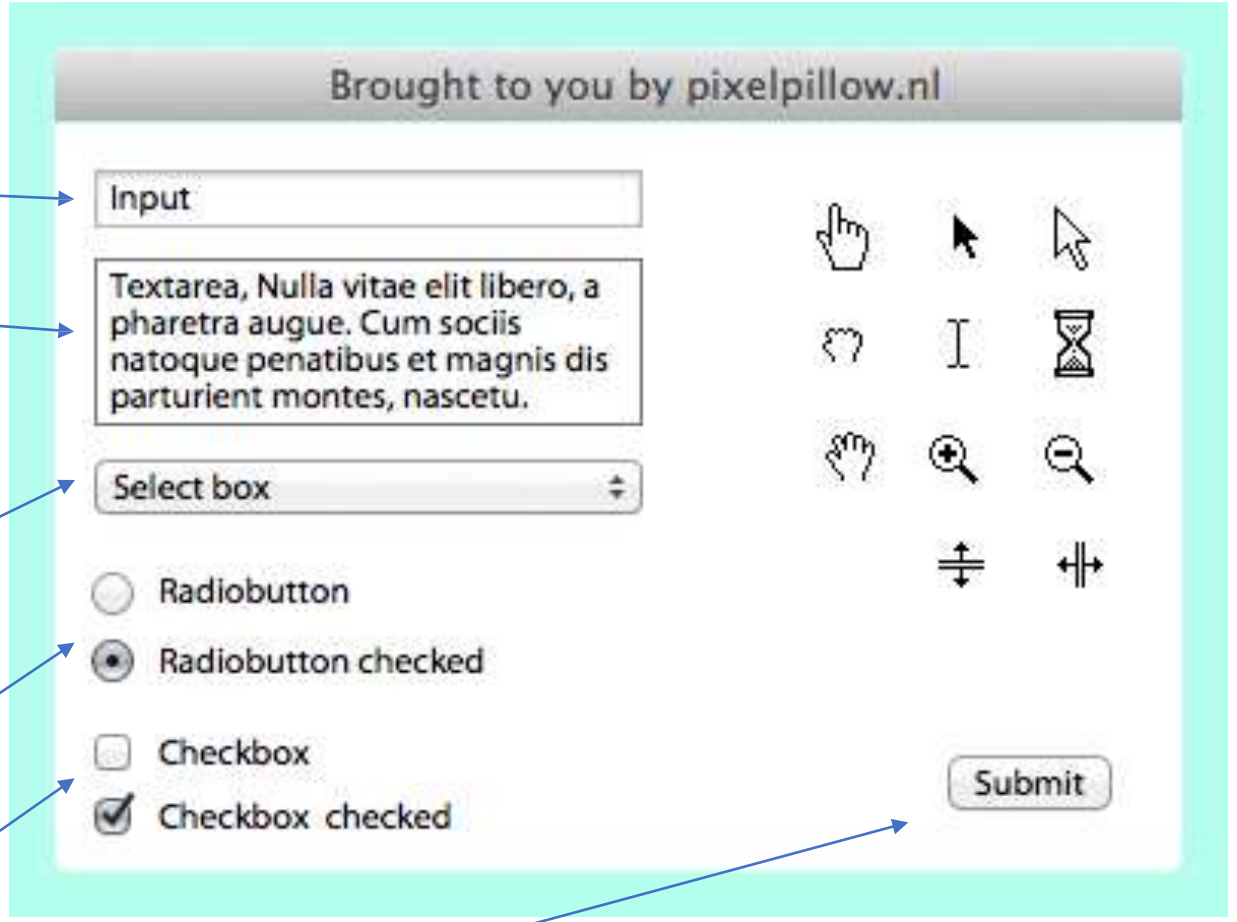
<TEXTAREA> </TEXTAREA>

<SELECT>
<option value='1'>1</option>
<option value='2'>2</option>
</SELECT>

<input type='radio'>

<input type='checkbox'>

<input type='submit '>



forms.py

Формы могут располагаться в любом месте проекта **Django**, но соглашение гласит, что их необходимо поместить в **`forms.py`** файл для каждого приложения.

В файле forms.py описывают взаимодействие с формой.

```
from django import forms
from .models import Post


class SampleModelForm(forms.ModelForm):
    class Meta:
        model = SampleModel
        fields = ('username',)
```

ModelForm – генератор формы с полями для Django

Класс Meta – внутренний метакласс Django

Метакласс в Django

Класс Meta используется для изменения поведения полей модели. Именно внутри этого класса создаётся экземпляр класса Django Model. В полях экземпляра перечисляются те поля модели, которые мы будем использовать.

id	login	username
1	timur@example.com	Тимур
2	masha@example.com	Маша
3	jonny@example.com	Джонни

```
class SampleModel(models.Model):
    login = models.CharField(max_length=100)
    username = models.CharField(max_length=50)
    def __str__(self):
        return self.username

class SampleModelForm(ModelForm):
    class Meta:
        model = SampleModel
        fields = ['username']
```

fields='__all__' чтобы включить все поля

Шаблоны вывода форм Django

{{ form.as_p }} – вывести форму как отдельный абзац

{{ form.as_ul }} - вывести форму как список

{{ form.as_table }} - вывести форму как таблицу

DJANGO requests

Django использует объекты запроса и ответа для передачи состояния через систему.

Атрибуты объекта request

httpRequest – http-запрос, обрабатываемый приложением Django. Атрибуты объекта:

- scheme: схема запроса (http или https)
- body: представляет тело запроса в виде строки байтов
- path: представляет путь запроса
- method: метод запроса (GET, POST, PUT и т.д.)
- encoding: кодировка
- content_type: тип содержимого запроса (значение заголовка CONTENT_TYPE)
- GET: объект в виде словаря, который содержит параметры запроса GET
- POST: объект в виде словаря, который содержит параметры запроса POST
- COOKIES: отправленные клиентом куки
- FILES: отправленные клиентом файлы
- META: хранит все доступные заголовки http в виде словаря.

GET и POST запросы

HTTP методы GET и POST используются для отправки данных на сервер.

Запрос GET передает данные в URL в виде пар "имя-значение".

Запрос POST передает данные в теле запроса.

Пример POST-запроса

Функция-обработчик post-запроса в файле views.py

```
def post_proc(request):
    form = PostForm()
    return render(request, 'some.html', {'form': form})
```

Теперь объект form может быть выведен в HTML-файле с помощью шаблонов вывода (см. слайд 5).

```
<form method="POST">
{% csrf_token %}
     {{ form.as_p }}
          <button type="submit">Save</button>
    </form>
```

Пример GET-запроса

views.py

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("<h2>Главная страница</h2>")
def user(request):
    age = request.GET.get("age")
    name = request.GET.get("name")
    return HttpResponse(f"<h2>Имя: {name}  Возраст: {age}</h2>")
```

urls.py

```
from django.urls import path
from main import views

urlpatterns = [
    path("", views.index),
    path("user/", views.user)
]
```

При обращении к приложению по адресу **http://127.0.0.1:8000/user/?name=Tom&age=22** параметр name будет иметь значение **"Tom"**, а параметр age - **22**.

Алгоритм добавления формы на сайт

1. urls.py - путь к приложению

```
path('*app_name*/', include('*app_name*.urls')),
```

2. main/templates/main/layout.html - ссылка на form.html

```
<a href="{%url 'create_form' %}">
```

3. *app_name*/ urls.py – путь к views

```
path('create_form', views.create, name='create_form')
```

4. *app_name*/ views.py – метод, создающий форму

```
•def index(request):
•   if request.method == 'POST':
•       form = SampleModelForm(request.POST)
•       if form.is_valid():
•           # Делайте что-то с валидными данными формы
•           form.save()
•           return redirect('mytest')
•           # Например, можно сохранить данные в базу данных
•   else:
•       form = SampleModelForm()
•
   return render(request, 'index.html', {'form':form})
```

forms.py – класс Meta, содержащий словарь widgets со всеми полями формы

```
class MyForm(ModelForm):  cla
   ss Meta:
         widgets = {"title": TextInput(attrs={
                  'class': 'form-control',
                  'placeholder': 'Название статьи'}),}
```

app_name/templates/*app_name*/create_form.html – добавить layout и саму форму

```
• <form method="POST">
• {% csrf_token %}
• {{ form.as_p }}
• <button type="submit">Save</button>
• </form>
```

Практика

Если ещё у вас нет формы , самое время её сделать!

•Вы должны создать веб-страницу, на которой пользователи могут отправлять свои отзывы и сообщения обратной связи. Для этого вам нужно:

•Создать модель **Feedback** с полями для имени пользователя, адреса электронной почты, темы сообщения и текста сообщения.

•Создать форму **FeedbackForm**, основанную на модели **Feedback**, с полями для всех атрибутов модели.

•Создать представление **feedback**, которое отображает форму обратной связи и обрабатывает данные, отправленные пользователем.

•Создать HTML-шаблон, который отображает форму обратной связи на веб-странице.

•Настроить маршрут (URL) для представления **feedback**.

•Добавить обработку успешной отправки формы и сообщение об успешной отправке на ваш HTML-шаблон.

Что почитать

Документация по Django-формам:
https://django.fun/ru/docs/django/4.1/topics/forms/modelforms/

Документация по полям формы (англ.)
https://docs.djangoproject.com/en/4.1/ref/forms/fields/

Руководство по ModelForms:
https://pythonim.ru/osnovy/klass-modelforms-v-django

Учебник по Django на примере создания блога (с реализацией GET и POST запросов) https://pocoz.gitbooks.io/django-v-primerah/content/

Путь от request до response:

https://django.fun/ru/articles/tutorials/put-ot-request-do-response-v-django/