Лабораторная работа №4.

Шаблоны Django. Отображение записей из БД на странице.

1. Что такое шаблон.

Шаблон — это файл, который ты можешь использовать повторно для отображения различной информации в заданном формате; например, ты можешь использовать шаблон, чтобы упростить написание письма, поскольку письма хоть и различаются по содержанию и получателю, но сохраняют общую структуру.

Шаблоны в Django написаны на языке HTML.

2. Создание простого шаблона

Создание шаблона означает создание файла шаблона. Шаблоны сохраняются в директории my_app/templates/my_app. Для начала создадим директорию templates внутри папки my_app. Затем создадим другую директорию my_app внутри папки templates:

```
my_app
└———templates
    └———my_app
```

Теперь создадим файл post_list.html внутри директории my_app/templates/my_app.

В файл post_list.html добавим следующий код:

```
<html>
    <head>
        <title>Прикладное ПО 1</title>
    </head>
```

```
        <body>
            <div>
                <h1><a href="/">Домашняя страница
</a></h1>
            </div>
            <div>
                <h2>Здравствуй, Пользователь!</h2>
            </div>
        </body>
    </html>
```

3. Добавление шаблона в представление

Теперь необходимо представлению, созданному в предыдущей лабораторной работе, указать шаблон.

Меняем код представления post_list в файле my_app/views.py:

```
def post_list(request):
        return render(request, 'my_app/post_list.html', {})
```

Теперь можно еще раз запустить сервер и открыть адрес 'http://127.0.0.1:8000/' и посмотреть результат рендера шаблона:

Домашняя страница

Здравствуй, Пользователь!

4. Передача объектов из модели в шаблон через представление.

Отдельные части уже размещены в нужных местах: модель Post определена в файле models.py, post_list — в файле views.py, и добавлен шаблон. Но как нам отобразить записи в шаблоне HTML-страницы? Ведь именно этого нам и нужно добиться: взять определенный контент (модели, сохранённые в базе данных) и аккуратно отобразить их в шаблоне.

Для этого и предназначены представления: соединять между собой модели и шаблоны. В post_list представлению нужно будет взять модели, которые мы хотим отобразить, и передать их шаблону. Таким образом, в представлениях мы определяем, что (какая модель) будет отображена в шаблоне.

Теперь нам нужно включить модель, которую мы определили в файле models.py в представление post_list. Добавим строку `from .models import Post` в файл my_app/views.py следующим образом:

```
from django.shortcuts import render
from .models import Post
```

Точка перед models означает текущую директорию или текущее приложение. Поскольку views.py и models.py находятся в одной директории, мы можем использовать точку . и имя файла (без расширения .py). Затем мы импортируем модель (Post).

Теперь мы хотим из модели получить список статей отсортированных по дате создания:

```
Post.objects.order_by('created_date')
```

Теперь нам нужно включить эту строку кода в файл my_app/views.py, добавив её в функцию def post_list(request):

```
from django.shortcuts import render
from .models import Post

def post_list(request):
    posts = Post.objects.order_by('created_date')
    return render(request, 'my_app/post_list.html', {})
```

Нам осталось передать QuerySet posts в шаблон. В функции render уже есть параметр request (т.е. всё, что мы получим от пользователя в качестве запроса через Интернет) и файл шаблона 'my_app/post_list.html'. Последний параметр, который выглядит как {}, — это место, куда мы можем добавить что-нибудь для использования в шаблоне. В итоге параметр будет выглядеть следующим образом: {'posts': posts}

В результате файл my_app/views.py должен выглядеть следующим образом:

```
from django.shortcuts import render
from .models import Post

def post_list(request):
    posts = Post.objects.order_by('created_date')
    return render(request, 'my_app/post_list.html',
{'posts': posts})
```

5. Отображение объектов в шаблоне.

Чтобы вставить переменную в шаблон Django, нам нужно использовать двойные фигурные скобки с именем переменной внутри: `{{ posts }}`

Для отображения всех записей в БД воспользуемся циклом `for`:

```
{% for post in posts %}
    {{ post }}
{% endfor %}
```

Мы также хотим, чтобы они отображались как статические записи HTML. Мы можем смешивать HTML и теги шаблонов. Наш элемент body будет выглядеть следующим образом (файл my_app/templates/my_app/post_list.html):

```
<html>
    <head>
        <title>Прикладное ПО 1</title>
    </head>
    <body>
        <div>
            <h1><a href="/">Домашняя страница</a></h1>
        </div>
        {% for post in posts %}
        <div>
            <p>created: {{ post.created_date }}</p>
            <h1><a href="">{{ post.title }}</a></h1>
            <p>{{ post.text|linebreaksbr }}</p>
        </div>
        {% endfor %}
```

```
    </body>
</html>
```

Всё, что находится между {% for %} и {% endfor %}, будет повторено для каждого объекта в списке.

После всех изменений при открытии адреса 'http://127.0.0.1:8000/' на странице будет отображаться список статей отсортированных по дате создания

Задание к лабораторной работе.

1. Создать шаблоны для отображения записей в БД по своему варианту.