**Методические указания по выполнению практического задания**

**Целью** занятия является практическое освоение теоретических знаний, полученных в процессе изучения темы 8 «Разработка онлайн-журнала (часть 1)».

**Задачи**, реализуемые на практическом занятии:

* Закрепление представления о понятиях, введенных на лекции.
* Получение практических навыков создания блога в среде Visual Studio 2017.

**Методические указания по выполнению практического задания 1**

1. Прежде чем приступить к выполнению задания 1, необходимо просмотреть видео, изучить презентацию, конспект и глоссарий к лекции.
2. Для закрепления представления о понятиях, введенных на лекции, найдите в конспекте к лекции понятия онлайн-журнал, блог и способы размещения контента на странице. По материалам лекции изучите структуру типичной статьи блога.
3. Для получения практических навыков создания блога в среде Visual Studio 2017:

* в конспекте к лекции найдите описание функций блога и популярных тем для блога, выберите тему для своего блога;
* в конспекте к лекции найдите описание способов размещения контента на странице;
* прочитайте теорию для практического занятия;
* выполните [задание 1](https://courses.openedu.ru/courses/course-v1:spbstu+WEBPYT+spring_2021/courseware/ad8be13d20c7444297348c709cb900b5/a63aa49cb1e64f5f934504461885b6ed/?child=first#yakor_1).

**Теория для практического занятия**

Для того, чтобы добавить на веб-сайт страницу со списком статей блога, необходимо:

1. В файл **urls.py** необходимо добавить шаблон URL для веб-страницы со списком постов.
2. В файл **views.py** необходимо добавить использование модели, которая была определена для блога в файле **models.py**. Для этого добавьте следующие строки в начале файла **views.py** в соответствующем блоке:

*from django.db import models*

*from .models import Blog*

3. В файл **views.py** добавить метод действия контроллера **blog** со следующим кодом:

*def blog(request):*

*"""Renders the blog page."""*

assert isinstance(request, HttpRequest)

*posts = Blog.objects.all() # запрос на выбор всех статей блога из модели*

*return render(*

*request,*

*'app/blog.html',*

*{*

*'title':'Блог',*

*'posts': posts, # передача списка статей в шаблон веб-страницы*

*'year':datetime.now().year,*

*}*

*)*

4. В папку **app\templates\app\** добавить файл **blog.html**(должен быть в кодировке UTF-8) для отображения списка статей блога со следующим кодом:

*{% extends "app/layout.html" %}*

*{% block content %}*

*<h2>{{ title }}</h2> <br />*

*{% for post in posts %}*

*<hr/>*

*<div class="">*

*<h2> {{post.title}} </h2>*

*<h4> {{post.posted}} </h4>*

*<p> {{post.description}} </p>*

*</div>*

*{% endfor %}*

*{% endblock %}*

5. В файле **templates\app\layout.html** добавьте в список пунктов меню пункт со ссылкой на блог.

Для того, чтобы добавить на веб-сайт страницу поста и реализовать на странице блога для каждой статьи ссылку на страницу поста (в виде кнопки «Читать далее» после краткого содержания статьи), необходимо:

1. В файл **urls.py** в шаблонах URL добавить строку:

*url(r'^(?P<parametr>\d+)/$', app.views.blogpost, name='blogpost'),*

2. В файл **blog.html** добавить строку (после строки вывода краткого содержания статьи) для перехода на страницу поста по ссылке «Читать далее»:

*<p><a href="{% url 'blogpost' parametr=post.id %}">* Читать далее*</a> </p>*

3. В файл **views.py** добавить код метода действия контроллера blogpost:

*def blogpost(request, parametr):*

*"""Renders the blogpost page."""*

assert isinstance(request, HttpRequest)

*post\_1 = Blog.objects.get(id=parametr) # запрос на выбор конкретной статьи по параметру*

*return render(*

*request,*

*'app/blogpost.html',*

*{*

*'post\_1': post\_1, # передача конкретной статьи в шаблон веб-страницы*

*'year':datetime.now().year,*

*}*

*)*

4. В папку app\templates\app\ добавить файл **blogpost.html** для отображения полного содержания конкретной статьи блога со следующим кодом:

*{% extends "app/layout.html" %}*

*{% block content %}*

*<h2> {{post\_1.title}} </h2>*

*<p> {{post\_1.content|linebreaksbr}} </p>      {# вывод текста по абзацам#}*

*{% endblock %}*

**Задание 1**

Цель задания – получить практические навыки создания блога в среде Visual Studio 2017.

1. Добавить в веб-проект Django страницу для отображения списка статей блога (постов) **в виде ленты** (например, **blog.html**) и проверить отображение списка тестовых постов на странице. Название страницы можно задать в соответствии с выбранной темой блога или «Блог» (см. материалы лекции 10 и теорию в подразделе [Теория для практического занятия](https://courses.openedu.ru/courses/course-v1:spbstu+WEBPYT+spring_2021/courseware/ad8be13d20c7444297348c709cb900b5/a63aa49cb1e64f5f934504461885b6ed/?child=first#08)). Необходимо использовать модель для статей блога, созданную в практическом задании по теме 7. Для каждого поста в списке отображать на странице заголовок, дату и краткое содержание.
2. Добавить в веб-проект Django страницу поста, содержащую заголовок и полное содержание статьи блога (например, **blogpost.html**). Реализовать на странице блога для каждой статьи ссылку «**Читать далее**» на страницу поста (после краткого содержания статьи) и проверить отображение каждого поста на странице. Определение конкретной статьи, отображаемой на странице, реализуется по передаваемому на сервер параметру **parameter**, равному уникальному идентификатору статьи (**post.id**) в модели (см. теорию в подразделе [Теория для практического занятия](https://courses.openedu.ru/courses/course-v1:spbstu+WEBPYT+spring_2021/courseware/ad8be13d20c7444297348c709cb900b5/a63aa49cb1e64f5f934504461885b6ed/?child=first#08)).
3. На страницу поста добавить ссылку «Назад» для перехода на веб-страницу блога.
4. В административном разделе заменить содержание тестовых статей на соответствующие выбранной теме. Проверить корректность отображения контента на страницах веб-сайта.
5. Используя стилевые таблицы реализовать индивидуальное оформление веб-сайта (на основе материалов темы 2 и справки библиотеки [BootStrap](https://webref.ru/layout/bootstrap" \t "_blank)). Применить индивидуальное оформление для следующих элементов веб-страниц сайта:
   * меню сайта;
   * заголовков;
   * текстов;
   * таблиц (если есть);
   * гиперссылок;
   * изображений.