**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка Интернет-приложений»

Отчет по лабораторной работе №4

«Шаблонизация»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-54Б |  | преподаватель каф. ИУ5 |
| Сафин Рустам |  | Гапанюк Ю. Е. |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2019 г.

**Описание задания лабораторной работы**

В этой ЛР вы создадите Django-проект, покажете пользователю статичную страницу, познакомитесь с конструкциями шаблонизаторов: переменные, теги, наследование шаблонов.

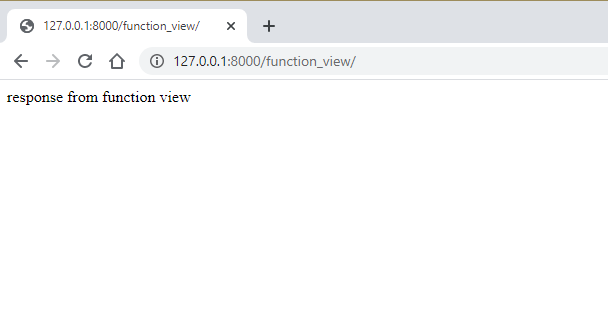
* Создать проект
* Реализовать view, в которых генерируются html-страницы
* В шаблонах должны быть использованы рассмотренные конструкции: переменные, вложенные значения, циклы, условия
* Все шаблоны должны расширять базовый шаблон
* Для элементов списка использовать тег include
* По нажатии на элемент списка должна открываться страница информации об элементе
* Для верстки необходимо использовать Bootstrap.

**Выполнение работы**

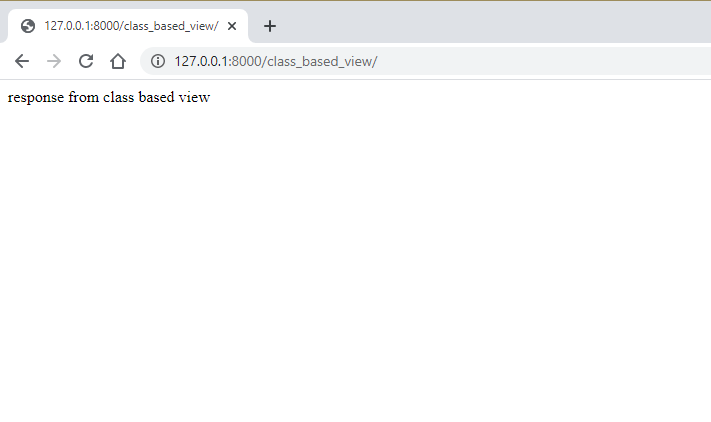
Для начала создадим Django-проект в PyCharm Professional Edition.

*Реализуем генераторы html-страниц в view: после запуска сервера вводим:*

При обращении к серверу по function\_view:



При обращении к серверу по class\_based\_view



Если использовать второй метод, то переопределяются методы 'get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'. Тогда при обращении по class\_based\_view будет вызван код метода, которым было произведено обращение.

Создаём базовый шаблон, от которого будут наследоваться все остальные:

<!DOCTYPE html>

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>{% block site %}{% endblock %}</title>**

**</head>**

**<body>**

**<div>Общая часть</div>**

**<div>**

**{% block body %}Переопределяется в наследниках{% endblock %}**

**</div>**

**</body>**

**</html>**

Для примера возьмём ситуацию, когда у нас есть несколько заказов, и для каждого из них нужно унаследовать и сгенерировать свой шаблон.

orders.html наследует base.html:

**{% extends 'base.html' %}**

**{% block title %}Заказы {% endblock %}**

**{% block body %}**

**<ul>**

**{% for order in orders %}**

**<li><a href="{% url 'order\_url' order.id %}">{{ order.title}}</a></li>**

**{% empty %}**

**<li>Пустой список заказов</li>**

**{% endfor %}**

**</ul>**

**{% endblock %}**

order.html генерируется свой для каждого заказа с индивидуальным номером:

**{% extends 'base.html' %}**

**{% block title %}Заказ № {{ order.id }}{% endblock %}**

**{% block body %}Страница заказа №{{ order.id }}{% endblock %}**

**Листинги модулей**

* asgi.py:

**"""**

**ASGI config for Lab4 project.**

**It exposes the ASGI callable as a module-level variable named ``application``.**

**For more information on this file, see**

**https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/**

**"""**

**import os**

**from django.core.asgi import get\_asgi\_application**

**os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Lab4.settings')**

**application = get\_asgi\_application()**

* settings.py:

**"""**

**Django settings for Lab4 project.**

**Generated by 'django-admin startproject' using Django 3.0.1.**

**For more information on this file, see**

**https://docs.djangoproject.com/en/3.0/topics/settings/**

**For the full list of settings and their values, see**

**https://docs.djangoproject.com/en/3.0/ref/settings/**

**"""**

**import os**

**# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)**

**BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))**

**# Quick-start development settings - unsuitable for production**

**# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/**

**# SECURITY WARNING: keep the secret key used in production secret!**

**SECRET\_KEY = '6$mh8wk9-+sauc764(4nwm6ig!gu=w84e#@ywicg=wv1+v8t5h'**

**# SECURITY WARNING: don't run with debug turned on in production!**

**DEBUG = True**

**ALLOWED\_HOSTS = []**

**# Application definition**

**INSTALLED\_APPS = [**

**'django.contrib.admin',**

**'django.contrib.auth',**

**'django.contrib.contenttypes',**

**'django.contrib.sessions',**

**'django.contrib.messages',**

**'django.contrib.staticfiles',**

**'lab4app.apps.Lab4AppConfig',**

**MIDDLEWARE = [**

**'django.middleware.security.SecurityMiddleware',**

**'django.contrib.sessions.middleware.SessionMiddleware',**

**'django.middleware.common.CommonMiddleware',**

**'django.middleware.csrf.CsrfViewMiddleware',**

**'django.contrib.auth.middleware.AuthenticationMiddleware',**

**'django.contrib.messages.middleware.MessageMiddleware',**

**'django.middleware.clickjacking.XFrameOptionsMiddleware',**

**]**

**ROOT\_URLCONF = 'Lab4.urls'**

**TEMPLATES = [**

**{**

**'BACKEND': 'django.template.backends.django.DjangoTemplates',**

**'DIRS': [os.path.join(BASE\_DIR, 'templates')]**

**,**

**'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',**

**],**

**},**

**},**

**]**

**WSGI\_APPLICATION = 'Lab4.wsgi.application'**

**# Database**

**# https://docs.djangoproject.com/en/3.0/ref/settings/#databases**

**DATABASES = {**

**'default': {**

**'ENGINE': 'django.db.backends.sqlite3',**

**'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),**

**}**

**}**

**# Password validation**

**# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators**

**AUTH\_PASSWORD\_VALIDATORS = [**

**{**

**'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',**

**},**

**{**

**'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',**

**},**

**{**

**'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',**

**},**

**{**

**'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',**

**},**

**]**

**# Internationalization**

**# https://docs.djangoproject.com/en/3.0/topics/i18n/**

**LANGUAGE\_CODE = 'en-us'**

**TIME\_ZONE = 'UTC**

**USE\_I18N = True**

**USE\_L10N = True**

**USE\_TZ = True**

**STATIC\_URL = '/static/'**

* **urls.py:**

**"""untitled URL Configuration**

**The `urlpatterns` list routes URLs to views. For more information please see:**

**https://docs.djangoproject.com/en/3.0/topics/http/urls/**

**Examples:**

**Function views**

**1. Add an import: from my\_app import views**

**2. Add a URL to urlpatterns: path('', views.home, name='home')**

**Class-based views**

**1. Add an import: from other\_app.views import Home**

**2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')**

**Including another URLconf**

**1. Import the include() function: from django.urls import include, path**

**2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))**

**"""**

**from django.contrib import admin**

**from django.urls import path**

**from django.conf.urls import url**

**from lab4app.views import function\_view**

**from lab4app.views import ExampleClassBased**

**from lab4app.views import BaseTemplateView**

**from lab4app.views import OrdersView**

**from lab4app.views import OrderView**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**url(r'^function\_view/', function\_view), # functional view**

**url(r'^class\_based\_view/', ExampleClassBased.as\_view()), # class based view**

**url(r'^base\_template', BaseTemplateView.as\_view()), # base template view**

**url(r'^order/(?P<id>\d+)', OrderView.as\_view(), name='order\_url')**

**]**

* **wsgi.py:**

**"""**

**WSGI config for Lab4 project.**

**It exposes the WSGI callable as a module-level variable named ``application``.**

**For more information on this file, see**

**https://docs.djangoproject.com/en/3.0/howto/deployment/wsgi/**

**"""**

**import os**

**from django.core.wsgi import get\_wsgi\_application**

**os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Lab4.settings')**

**application = get\_wsgi\_application()**

* **admin.py:**

**from django.contrib import admin**

**# Register your models here.**

* **apps.py:**
* **models.py:**

**from django.db import models**

**# Create your models here.**

* **tests.py:**

**from django.db import models**

**# Create your models here.**

* **views.py**

**from django.shortcuts import render**

**# Create your views here.**

**from django.views.generic import View**

**from django.http import HttpResponse**

**def function\_view(request):**

**return HttpResponse('response from function view')**

**class ExampleClassBased(View):**

**def get(self, request):**

**return HttpResponse('response from class based view')**

**class BaseTemplateView(View): # простая страница**

**def get(self, request):**

**return render(request, 'base.html')**

**class OrdersView(View):**

**def get(self, request):**

**data = {**

**'orders': {**

**{'title': 'Первый заказ', 'id': 1},**

**{'title': 'Второй заказ', 'id': 2},**

**{'title': 'Третий заказ', 'id': 3}**

**}**

**}**

**return render(request, 'orders.html', data)**

**class OrderView(View):**

**def get(self, request):**

**data = {**

**'order': {**

**'id': id**

**}**

**}**

**return render(request, 'order.html', data)**