ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № _10_

Дисциплина: Backend-разработка

Тема: Создание простого веб-приложения на Django

Выполнил(а): студент(ка) группы __221-379__

_____Кодиров Жамшид Мурод угли_____
(Фамилия И.О.)

Проверил: _______ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания:_____

_____

_____

Москва

2025

Задание

Цель работы: Получить практические навыки работы с фреймворком Django: установка и настройка, создание проекта и приложения, работа с представлениями, маршрутизацией, обработкой запросов и отправкой ответов.

Задание:

Часть 1: Настройка проекта и создание приложения

Установка Django:

Установите Django на свою локальную машину с помощью pip.

Проверьте установку, выполнив команду django-admin --version.

Создание проекта:

Создайте новый проект Django с именем my_first_django_project.

Перейдите в директорию проекта и запустите сервер разработки командой python manage.py runserver.

Откройте браузер и убедитесь, что сервер работает, перейдя по адресу http://127.0.0.1:8000/.

Создание приложения:

Создайте новое приложение с именем my_first_app внутри проекта.

Зарегистрируйте приложение в файле settings.py проекта.

Часть 2: Работа с представлениями и маршрутизацией

Создание простого представления:

В файле views.py приложения my_first_app создайте функцию-представление hello_world, которая возвращает простой HTML-ответ: <h1>Hello, World!</h1>.

Создайте маршрут для этого представления в файле urls.py приложения, используя функцию path.

Проверьте работу представления, перейдя по соответствующему URL в браузере.

Обработка параметров запроса:

Измените представление hello_world, чтобы оно принимало параметр name из URL и возвращало ответ <h1>Hello, {name}!</h1>.

Создайте маршрут с параметром в файле urls.py, используя функцию path или re_path.

Проверьте работу представления, перейдя по URL с разными значениями параметра name.

Получение данных из строки запроса:

Измените представление, чтобы оно также принимало параметр age из строки запроса и возвращало ответ <h1>Hello, {name}! You are {age} years old.</h1>.

Проверьте работу представления, перейдя по URL с параметрами в строке запроса.

Переадресация и отправка статусных кодов:

Создайте новое представление redirect_example, которое перенаправляет пользователя на представление hello_world с параметрами по умолчанию.

Используйте функцию redirect для перенаправления и HttpResponse для отправки статусного кода 302.

Проверьте работу переадресации, перейдя по соответствующему URL.

Отправка JSON-ответа:

Создайте новое представление json_example, которое возвращает JSON-ответ с данными о пользователе (например, имя и возраст).

Используйте функцию JsonResponse для отправки JSON-ответа.

Проверьте работу представления, перейдя по соответствующему URL и просмотрев ответ в инструментах разработчика браузера.

Работа с куками:

Измените представление hello_world, чтобы оно устанавливало куку с именем пользователя.

Создайте новое представление show_cookies, которое отображает все куки, установленные в браузере.

Проверьте работу представлений, перейдя по соответствующим URL и просмотрев куки в инструментах разработчика браузера.

Часть 3: Вложенные маршруты и функция include

Создание вложенных маршрутов:

Создайте новое приложение nested_app и зарегистрируйте его в проекте.

В приложении nested_app создайте несколько представлений и маршрутов.

Используйте функцию include в файле urls.py проекта, чтобы включить маршруты из приложения nested_app в общий маршрут проекта.

Проверьте работу вложенных маршрутов, перейдя по соответствующим URL.

Ход работы:

1. **Установка Django:**

   В командной строке выполнили установку Django с помощью pip:

   ```
   pip install django
   ```

   Проверили успешность установки командой:

   ```
   django-admin --version
   ```

   После этого убедились, что Django установлен и готов к работе (выводится номер установленной версии).

2. **Создание проекта:**

   В терминале выполнили команду для создания нового проекта Django:

   ```
   django-admin startproject my_first_django_project
   ```

   Перешли в директорию проекта:

   ```
   cd my_first_django_project
   ```

   Запустили сервер разработки:

   ```
   python manage.py runserver
   ```

Перешли в браузер и открыли адрес http://127.0.0.1:8000/, чтобы убедиться, что сервер работает. Появилась стандартная стартовая страница Django.

3. **Создание приложения:**

   В терминале выполнили:

   ```
   python manage.py startapp my_first_app
   ```

   В открывшемся каталоге приложения появились файлы: `views.py`, `models.py`, `admin.py`, `apps.py` и другие.

   Приложение зарегистрировали в файле `settings.py` проекта, добавив его в список `INSTALLED_APPS`:

   ```
   INSTALLED_APPS = [
       ...
       'my_first_app',
   ]
   ```

4. **Создание простого представления:**

   В файле `my_first_app/views.py` создали функцию:

   ```
   from django.http import HttpResponse

   def hello_world(request):
       return HttpResponse("<h1>Hello, World!</h1>")
   ```

   Затем в корне приложения создали файл `urls.py` (если он отсутствует) и добавили маршруты:

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

   urlpatterns = [
       path('hello/', views.hello_world),
   ]
   ```

   В файле `my_first_django_project/urls.py` подключили маршруты приложения:

   ```
   from django.contrib import admin
   ```

```
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('my_first_app.urls')),
]
```

После этого открыли http://127.0.0.1:8000/hello/ и увидели сообщение: `Hello, World!`.

5. **Обработка параметров URL:**

   Модифицировали функцию `hello_world`, чтобы она принимала параметр `name`:

   ```
   def hello_world(request, name='World'):
       return HttpResponse(f"<h1>Hello, {name}!</h1>")
   ```

   Обновили маршруты:

   ```
   path('hello/<str:name>/', views.hello_world),
   ```

   Проверили результат по адресу, например:
   http://127.0.0.1:8000/hello/Alex/ → `Hello, Alex!`.

6. **Получение данных из строки запроса:**

   Изменили `hello_world`, добавив чтение параметра `age`:

   ```
   def hello_world(request, name='World'):
       age = request.GET.get('age', 'unknown')
       return HttpResponse(f"<h1>Hello, {name}! You
   are {age} years old.</h1>")
   ```

   Перешли по адресу, например: http://127.0.0.1:8000/hello/Alex/?age=22

7. **Переадресация (редирект):**

   Создали представление `redirect_example`, которое выполняет редирект:

   ```
   from django.shortcuts import redirect

   def redirect_example(request):
       return redirect('/hello/Guest/?age=25')
   ```

   Добавили маршрут:

```
path('redirect/', views.redirect_example),
```

Проверили по адресу: http://127.0.0.1:8000/redirect/

8. **Отправка JSON-ответа:**

   Создали представление `json_example`:

```
from django.http import JsonResponse

def json_example(request):
    data = {'name': 'Alice', 'age': 30}
    return JsonResponse(data)
```

   Добавили маршрут:

```
path('json/', views.json_example),
```

   Проверили по адресу: http://127.0.0.1:8000/json/. В браузере отобразился JSON-объект.

9. **Работа с куками:**

   Изменили `hello_world`, чтобы установить куку `username`:

```
def hello_world(request, name='World'):
    age = request.GET.get('age', 'unknown')
    response = HttpResponse(f"<h1>Hello, {name}!
You are {age} years old.</h1>")
    response.set_cookie('username', name)
    return response
```

   Создали новое представление `show_cookies`:

```
def show_cookies(request):
    return
HttpResponse(f"<pre>{request.COOKIES}</pre>")
```

   Добавили маршрут:

```
path('cookies/', views.show_cookies),
```

   Проверили по адресу: http://127.0.0.1:8000/cookies/ — отобразились установленные куки.

10. **Создание вложенных маршрутов:**

Создали новое приложение:

```
python manage.py startapp nested_app
```

Добавили его в `INSTALLED_APPS`:

```
'nested_app',
```

В `nested_app/views.py` создали два представления:

```
def nested_home(request):
    return HttpResponse("<h2>This is nested app
home.</h2>")

def nested_about(request):
    return HttpResponse("<h2>About nested app.</h2>")
```

В `nested_app/urls.py`:

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

urlpatterns = [
    path('', views.nested_home),
    path('about/', views.nested_about),
]
```

В `my_first_django_project/urls.py` добавили:

```
path('nested/', include('nested_app.urls')),
```

Проверили по адресам:

- http://127.0.0.1:8000/nested/
- http://127.0.0.1:8000/nested/about/

Теперь можно переходить по маршрутам:

- http://127.0.0.1:8000/hello/Alice/?age=30
- http://127.0.0.1:8000/redirect/
- http://127.0.0.1:8000/json/
- http://127.0.0.1:8000/cookies/
- http://127.0.0.1:8000/nested/
- http://127.0.0.1:8000/nested/about/

Вывод:

В ходе лабораторной работы был пошагово реализован полноценный Django-проект с двумя приложениями. Мы изучили:

- установку Django и запуск проекта;
- создание приложений и их регистрацию;
- создание различных представлений с поддержкой URL-параметров и строки запроса;
- перенаправления пользователей с использованием `redirect`;
- отправку JSON-ответов через `JsonResponse`;
- работу с куками: установка и чтение;
- организацию маршрутов с использованием функции `include` и реализацию вложенной маршрутизации.

Результатом работы стал функциональный Django-проект, содержащий набор представлений и маршрутов, демонстрирующих базовые возможности фреймворка. Все части задания были выполнены успешно, представления работают корректно, маршруты обрабатываются согласно логике, заданной в коде.

Ссылка на Github: https://github.com/QodirovJM/BackendPython