# Лабораторная работа: Работа с представлениями в Django

## Цель работы:

Научиться создавать представления (views) в Django, работать с функциями и классами представлений, а также передавать данные в шаблоны для их отображения на веб-странице.

## Описание задания:

Создать простое Django-приложение, включающее представления для отображения страниц, обработки данных, и использования различных типов представлений (функциональных и классовых). Также настроить маршрутизацию URL для соединения представлений с URL-адресами.

## Инструменты:

- Установленная среда Python
- Установленный Django
- Текстовый редактор (например, PyCharm, Visual Studio Code)

## Ход работы:

### 1. Создание нового проекта Django:

Создайте новый проект Django, используя команду:
```bash
django-admin startproject myproject
cd myproject
```
Создайте новое приложение, например `myapp`, командой:
```bash
python manage.py startapp myapp
```
Добавьте `myapp` в список `INSTALLED\_APPS` в `settings.py`:
```python
# myproject/settings.py
INSTALLED\_APPS = [
 ...,
 'myapp',
]
```

### 2. Создание простого функционального представления:

В `views.py` приложения `myapp` создайте простое представление, которое возвращает текст.
```python
# myapp/views.py
from django.http import HttpResponse

def hello\_view(request):
 return HttpResponse("Hello, this is a functional view!")
```

### 3. Настройка маршрута URL для функционального представления:

Создайте файл `urls.py` в приложении `myapp` и добавьте туда маршрут для `hello\_view`.
Включите `myapp/urls.py` в основной файл `urls.py` проекта.
```python
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
 path('hello/', views.hello\_view, name='hello'),
]

# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

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

### 4. Создание представления с использованием шаблона:

Создайте директорию `templates/myapp` и добавьте туда файл `home.html`.
Внутри `home.html` добавьте простой HTML-код, чтобы отобразить заголовок и переменные из представления:
```html
<!-- templates/myapp/home.html -->
<h1>Добро пожаловать на домашнюю страницу!</h1>
<p>Сегодня: {{ today }}</p>
```
В `views.py` создайте представление, которое передает данные в шаблон:
```python
# myapp/views.py
from django.shortcuts import render
from datetime import date

def home\_view(request):
 today = date.today()
 return render(request, 'myapp/home.html', {'today': today})
```
Обновите `urls.py`, чтобы добавить маршрут для `home\_view`:
```python
# myapp/urls.py
urlpatterns = [
 path('hello/', views.hello\_view, name='hello'),
 path('home/', views.home\_view, name='home'),
]
```

### 5. Создание классового представления:

Используйте встроенные классы представлений для обработки отображений. В `views.py` создайте простое классовое представление:
```python
# myapp/views.py
from django.views.generic import TemplateView

class AboutView(TemplateView):
 template\_name = "myapp/about.html"
```
Создайте шаблон `about.html` и добавьте простой HTML-контент:
```html
<!-- templates/myapp/about.html -->
<h1>О нас</h1>
<p>Это страница "О нас", созданная с использованием классового представления.</p>
```
Добавьте маршрут для `AboutView` в `urls.py`:
```python
# myapp/urls.py
from .views import AboutView

urlpatterns = [
 path('hello/', views.hello\_view, name='hello'),
 path('home/', views.home\_view, name='home'),
 path('about/', AboutView.as\_view(), name='about'),
]
```

### 6. Проверка работы:

Запустите сервер с помощью команды:
```bash
python manage.py runserver
```
Откройте браузер и перейдите по URL-адресам `/hello/`, `/home/`, и `/about/`, чтобы убедиться, что представления работают корректно.

## Контрольные вопросы:

- Что такое представление в Django и для чего оно используется?

- В чем отличие функциональных и классовых представлений в Django?

- Как передать данные из представления в шаблон?

- Какой метод используется для обработки GET-запросов в классовых представлениях?

## Дополнительное задание:

Создайте форму в Django для сбора данных от пользователя, обработайте ее в представлении и отобразите введенные данные на новой странице, используя как функциональные, так и классовые представления.