# Лабораторная работа: Передача данных из базы данных на веб-страницы Django

## Цель работы:

Научиться извлекать данные из базы данных и отображать их на веб-страницах Django с использованием моделей, представлений и шаблонов.

## Описание задания:

Создать Django-приложение с базой данных, реализовать модель данных, извлечь данные из модели и передать их в шаблон для отображения на веб-странице.

## Инструменты:

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

## Ход работы:

### 1. Создание проекта и приложения 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 модели данных:

В приложении `myapp` откройте файл `models.py` и создайте модель `Article`, представляющую информацию о статьях, такие как `title`, `content`, `author`, и `published\_date`:
```python
# myapp/models.py
from django.db import models

class Article(models.Model):
 title = models.CharField(max\_length=200)
 content = models.TextField()
 author = models.CharField(max\_length=100)
 published\_date = models.DateField()

 def \_\_str\_\_(self):
 return self.title
```
Выполните миграции для создания таблицы `Article` в базе данных:
```bash
python manage.py makemigrations
python manage.py migrate
```

### 3. Добавление модели в админ-панель:

Зарегистрируйте модель `Article` в `admin.py` для управления записями через интерфейс администратора:
```python
# myapp/admin.py
from django.contrib import admin
from .models import Article

admin.site.register(Article)
```
Создайте суперпользователя для доступа в админ-панель:
```bash
python manage.py createsuperuser
```
Запустите сервер, войдите в админ-панель и добавьте несколько записей для модели `Article`.

### 4. Создание представления для передачи данных в шаблон:

В `views.py` создайте представление `article\_list`, которое передает список всех статей из базы данных в шаблон:
```python
# myapp/views.py
from django.shortcuts import render
from .models import Article

def article\_list(request):
 articles = Article.objects.all()
 return render(request, 'myapp/article\_list.html', {'articles': articles})
```

### 5. Создание шаблона для отображения данных:

Создайте директорию `templates/myapp` и добавьте туда файл `article\_list.html`.
Внутри `article\_list.html` отобразите информацию о каждой статье, используя переданные данные:
```html
<!-- templates/myapp/article\_list.html -->
<h1>Список статей</h1>
<ul>
 {% for article in articles %}
 <li>
 <h2>{{ article.title }}</h2>
 <p><strong>Автор:</strong> {{ article.author }} | <strong>Дата:</strong> {{ article.published\_date }}</p>
 <p>{{ article.content|truncatewords:20 }}</p>
 <a href="{% url 'article\_detail' article.id %}">Читать далее</a>
 </li>
 {% endfor %}
</ul>
```

### 6. Создание представления для отображения подробной информации:

В `views.py` создайте представление `article\_detail`, которое будет отображать данные конкретной статьи по её `id`:
```python
# myapp/views.py
from django.shortcuts import render, get\_object\_or\_404
from .models import Article

def article\_detail(request, pk):
 article = get\_object\_or\_404(Article, pk=pk)
 return render(request, 'myapp/article\_detail.html', {'article': article})
```

### 7. Создание шаблона для отображения подробной информации:

Создайте файл `article\_detail.html` в папке `templates/myapp` для отображения содержимого конкретной статьи:
```html
<!-- templates/myapp/article\_detail.html -->
<h1>{{ article.title }}</h1>
<p><strong>Автор:</strong> {{ article.author }}</p>
<p><strong>Дата публикации:</strong> {{ article.published\_date }}</p>
<div>
 <p>{{ article.content }}</p>
</div>
<a href="{% url 'article\_list' %}">Назад к списку статей</a>
```

### 8. Добавление маршрутов URL для представлений:

В `urls.py` создайте маршруты для `article\_list` и `article\_detail`, чтобы можно было получить к ним доступ через URL-адреса:
```python
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
 path('articles/', views.article\_list, name='article\_list'),
 path('articles/<int:pk>/', views.article\_detail, name='article\_detail'),
]

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

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

### 9. Проверка работы:

Запустите сервер и откройте страницу `/articles/` в браузере, чтобы увидеть список статей.
Кликните на ссылку «Читать далее» под статьёй, чтобы перейти на страницу с её полным содержимым.

## Контрольные вопросы:

- Как передать данные из базы данных на страницу в Django?

- Чем отличается представление для списка записей и представление для отображения одной записи?

- Как использовать фильтрацию для получения конкретной записи в базе данных?

- Что делает функция `get\_object\_or\_404`, и в каких случаях её полезно использовать?

## Дополнительное задание:

Создайте форму для добавления новых статей и настройте представление, чтобы передавать данные из формы в модель `Article`.