МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Поволжский государственный университет телекоммуникаций и информатики»

Кафедра «Программная инженерия» (ПрИ)
(наименование кафедры)

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Разработка серверной части web-приложений

(наименование учебной дисциплины)

Для направления подготовки бакалавров: 09.03.01 Информатика и вычислительная техника
(код и наименование направления подготовки)

Самара
2025

Лабораторная работа № 1

1. Введение в Django

Django — это высокоуровневый веб-фреймворк на языке Python, который позволяет быстро разрабатывать безопасные и поддерживаемые веб-приложения. Он следует принципам DRY (Don't Repeat Yourself — "Не повторяйся") и Convention Over Configuration ("Соглашения вместо конфигураций").

Основные преимущества Django:

- Быстрая разработка: Django предоставляет множество встроенных инструментов для создания веб-приложений.
- Безопасность: Django включает встроенные механизмы для защиты от распространенных уязвимостей, таких как SQL-инъекции, XSS и CSRF.
- Масштабируемость: Django подходит для проектов любого масштаба, от небольших сайтов до крупных порталов.
- Поддержка ORM: Django предоставляет объектно-реляционное отображение (ORM) для работы с базами данных.

2. Архитектура Django

Django следует архитектуре **MVT (Model-View-Template)**:

- **Model (Модель):** Отвечает за работу с данными. Модели описывают структуру базы данных и взаимодействуют с ней через ORM.
- **View (Представление):** Обрабатывает логику приложения. Представления получают запросы от клиента, обрабатывают их и возвращают ответ.
- **Template (Шаблон):** Отвечает за отображение данных. Шаблоны используются для генерации HTML-кода, который отправляется клиенту.

3. Основные компоненты Django

Проект (Project):

- Проект — это совокупность настроек и приложений, которые вместе образуют веб-сайт.
- Команда для создания проекта: django-admin startproject myproject.

Приложение (App):

- Приложение — это модуль, который выполняет конкретную функцию (например, блог, магазин, аутентификация).
- Команда для создания приложения: python manage.py startapp myapp.

Маршрутизация (URL routing):

- Маршруты определяют, какие представления будут обрабатывать запросы по определенным URL.
- Маршруты настраиваются в файле urls.py.

Представления (Views):

- Представления — это функции или классы, которые обрабатывают запросы и возвращают ответы.
- Пример функции-представления:

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Привет, мир!")
```

Шаблоны (Templates):

- Шаблоны используются для генерации HTML-кода. Они могут включать переменные, теги и фильтры.
- Пример шаблона:

```
<h1>{{ title }}</h1>
<p>{{ content }}</p>
```

Модели (Models):

- Модели описывают структуру базы данных. Каждая модель соответствует таблице в базе данных.
- Пример модели:

```
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    pub_date = models.DateTimeField(auto_now_add=True)
```

Админ-панель (Admin):

- Админ-панель — это встроенный интерфейс для управления данными в базе данных.
- Чтобы зарегистрировать модель в админ-панели, используйте:

```
from django.contrib import admin
from .models import Article

admin.site.register(Article)
```

4. Работа с базами данных

Django использует ORM для работы с базами данных. ORM позволяет взаимодействовать с базой данных, используя Python-код вместо SQL-запросов.

Основные операции с моделями:

- Создание записи:

```
article = Article(title="Заголовок", content="Текст статьи")
article.save()
```

- Чтение записей:

```
articles = Article.objects.all()  # Все записи
article = Article.objects.get(id=1)  # Запись по ID
```

Обновление записей:

```
article = Article.objects.get(id=1)
article.title = "Новый заголовок"
article.save()
```

Удаление записей:

```
article = Article.objects.get(id=1)
article.delete()
```

5. Формы в Django

Формы используются для сбора данных от пользователей. Django предоставляет два основных способа работы с формами:

1. **Ручное создание форм:** Используйте HTML-формы и обрабатывайте данные в представлениях.
2. **Django Forms:** Встроенный механизм для создания и валидации форм.

Пример формы:

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(label="Ваше имя")
    email = forms.EmailField(label="Ваш email")
    message = forms.CharField(label="Сообщение", widget=forms.Textarea)
```

Обработка формы в представлении:

```
from django.shortcuts import render, redirect
from .forms import ContactForm

def contact(request):
```

```
    if request.method == "POST":
        form = ContactForm(request.POST)
        if form.is_valid():
            # Обработка данных
            return redirect("success")
    else:
        form = ContactForm()
    return render(request, "contact.html", {"form": form})
```

6. Статические файлы

Статические файлы (CSS, JavaScript, изображения) хранятся в директории static. Чтобы подключить статические файлы в шаблоне, используйте:

```
{% load static %}
<link rel="stylesheet" href="{% static 'myapp/styles.css' %}">
```

7. Аутентификация и авторизация

Django предоставляет встроенную систему аутентификации и авторизации. Основные функции:

- Регистрация и вход пользователей.
- Ограничение доступа к представлениям.
- Управление группами и разрешениями.

Пример ограничения доступа:

```
from django.contrib.auth.decorators import login_required


@login_required
def profile(request):
    return render(request, "profile.html")
```

8. Дополнительные возможности Django

- **Middleware:** Промежуточное ПО для обработки запросов и ответов.
- **Кэширование:** Ускорение работы приложения за счет кэширования данных.
- **Международization (i18n):** Поддержка многоязычных сайтов.
- **Django REST Framework (DRF):** Создание API для веб-приложений.

1.1Введение в Django

Цель работы

Ознакомиться с базовыми понятиями фреймворка Django, его структурой и принципами работы. Освоить настройку среды разработки. Создание проекта и маршрутизации.

Задания:

Для работы с Django необходимо использовать среду разработки Python!

1. Установите Django и создайте новый проект
2. Настройте маршрутизацию для страницы с приветственным сообщением.
3. Создайте базовый HTML-шаблон для вывода сообщения «Hello, world!».

Контрольные вопросы

1. Основные понятия Django.
2. Как работает маршрутизация в Django?
3. Как подключаются HTML-шаблоны?

1.2 Модели и работа с базой данных

Цель работы

Научиться создавать модели в Django, использовать встроенную ORM для работы с базами данных и выполнять миграции.

Задания:

1. Создайте модель «Article» с полями: заголовок, содержимое, дата публикации.
2. Настройте базу данных и выполните миграции.
3. Добавьте модель в Django Admin для управления записями.
4. Реализуйте вывод всех статей на отдельной странице.

Контрольные вопросы

1. Что такое ORM и как она работает в Django?
2. Как создать и применить миграции?
3. Какие типы полей существуют в Django моделях?
4. Как работать с Admin Django?

1.3 Формы и валидация данных

Цель работы:

Освоить работу с формами в Django, их создание, обработку и валидацию данных.

Задания:

1. Создайте форму для добавления новых статей (заголовок, содержимое).
2. Реализуйте маршруты для отображения формы и обработки данных формы.
3. Настройте валидацию данных формы (например, проверка обязательных полей).
4. Реализуйте редактирование существующих статей через форму.

Контрольные вопросы:

1. Как создать и обработать форму в Django?
2. В чем отличие между классами Form и ModelForm?
3. Как реализовать валидацию данных в Django?
4. Как настроить маршруты для обработки POST-запросов?

1.4 Аутентификация и авторизация

Цель работы

Изучить систему аутентификации и авторизации в Django, настроить регистрацию пользователей и управление доступом к страницам.

Задания:

1. Реализуйте систему регистрации и авторизации пользователей.
2. Ограничьте доступ к созданию и редактированию статей только для авторизованных пользователей.
3. Настройте перенаправление на страницу входа при попытке доступа к закрытым маршрутам.
4. Реализуйте функционал сброса пароля через e-mail.

Контрольные вопросы

1. Как настроить систему аутентификации в Django?
2. Как ограничить доступ к определенным маршрутам для неавторизованных пользователей?
3. Как работает middleware в Django и как его использовать для авторизации?
4. Как реализовать функционал сброса пароля?

1.5 Создание REST API с использованием Django Rest Framework (DRF)

Цель работы

Научиться работать с Django Rest Framework для создания REST API, реализовать CRUD операции через API.

Задания

1.5.1 Установите и настройте Django Rest Framework.

1.5.2 Создайте API для сущности «Article» с поддержкой операций создания, чтения, обновления и удаления.

1.5.3 Реализуйте сериализацию данных и маршрутизацию для API.

1.5.4 Ограничьте доступ к API методам для авторизованных пользователей.

Контрольные вопросы

1. Что такое Django Rest Framework и зачем его использовать?
2. Как реализовать сериализаторы и ViewSets в DRF?
3. Как реализовать аутентификацию и авторизацию для API?
4. Как настроить маршрутизацию для API в Django?

1.6 Тестирование Django-приложений

Цель работы

Научиться работать с Django Rest Framework для создания REST API, реализовать CRUD операции через API.

Задания

1. Напишите юнит-тесты для модели «Article».
2. Реализуйте тесты для проверки правильности работы форм.
3. Напишите тесты для проверки работы API (создание, чтение, обновление и удаление статей).
4. Настройте тестирование обработки ошибок (например, доступ к несуществующим страницам).

Контрольные вопросы

1. Какие виды тестирования поддерживает Django?
2. Как создать тесты для моделей и форм?
3. Как тестировать API с помощью Django TestCase?
4. Какие ошибки могут возникать и как их отслеживать с помощью тестов?