Заключение. В работе исследован процесс рассеяния электромагнитных волн цилиндрической частицей в изотропной среде при облучении частицы плоской волной, направленной нормально к продольной оси симметрии цилиндра. Основным результатом является сведение системы исходных уравнений в частных производных к системе алгебраических уравнений. Результаты, полученные в ходе исследования рассеяния волн биизотропными цилиндрами, могут быть полезны при расчете полей, излучаемых стержневыми антеннами, а также при исследовании волноводов и рассеивателей.

Литература

1. Lindell, I. V. Electromagnetic waves in chiral and bi-isotropic media/ I. V, Lindell, A. H. Sihvola, S. A. Tretyakov, A. J. Viitanen. – Artech House, Boston and London. – 1994. – 344 P.

2. Barabas, M. Scattering of a plane wave by radially stratified titled cylinder / M. Barabas. // J. Opt. Soc. Amer. A 4, 1987. — Pp. 2240–2248.

3. Chittayil, K. Electromagnetic scattering by a chiral cylinder immersed in another chiral medium / K. Chittayil, A. Lakhtakia // Optik 89, 1991. – Pp. 59–64.

Н. А. Доломакин
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **С. А. Лукашевич**, ст. преподаватель

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ НА ЯЗЫКЕ PYTHON

Паттерн проектирования предоставляет эффективный инструмент для документирования принимаемых решений, основанных на предыдущем опыте и извлеченных уроках. Этот подход обеспечивает структурирование процесса реализации модели проектирования с использованием нескольких программных компонентов. В результате модель способствует ускорению взаимодействия между различными компонентами, что облегчает слаженную работу [1].

Применение шаблонов проектирования приносит множество выгод в процессе разработки программного обеспечения. В первую очередь, они способствуют ускорению процесса создания приложений. Шаблоны предоставляют готовые, оптимизированные решения для типичных задач, что позволяет избежать необходимости разработки новых компонентов с нуля. Это существенно сокращает время, затрачиваемое на проектирование и кодирование [1].

В рамках данной статьи будут рассмотрены два шаблона проектирования, будут показаны способы их реализации в связке с веб-фреймворком Django языка Python. Это будет тестовая реализация шаблонов в рамках проекта, которая покажет, как шаблон может быть интегрирован даже в готовое приложение. Конечно, данная реализация не имеет смысл в рамках разработки коммерческих проектов, однако необходимо знать эти шаблоны проектирования и как их применять на практике.

Сначала рассмотрим шаблон проектирования “Команда”. Данный паттерн представляет собой поведенческий паттерн, который инкапсулирует запрос в виде объекта, позволяя параметризовать клиентов с различными запросами, ставить запросы в очередь, а также поддерживать отмену операций [2]. Основные преимущества использования данного шаблона проектирования заключаются в следующих пунктах:

1. Отделение отправителя от получателя: Шаблон позволяет отделить объект, инициирующий запрос от объекта, который фактически выполняет операцию. Это уменьшает связанность и делает систему более гибкой, так как отправитель не зависит напрямую от реализации получателя.

2. Поддержка отмены и повтора операций: Команды могут быть сохранены в истории, что дает возможность отменять и повторять операции. Это полезно, например, при реализации систем отмены действий в графических редакторах [2].

3. Создание очереди команд: Команды могут быть добавлены в очередь, что позволяет эффективно управлять последовательностью выполнения операций.

4. Улучшение расширяемости системы: Добавление новых команд и их изменение можно выполнять независимо от клиентского кода.

Пример реализации данного шаблона проектирования в рамках разработки веб-приложения “Афиша” показан на рисунке 1.

Command

```python
# Модель мероприятия
from django.db import models

class Event(models.Model):
    name = models.CharField(max_length=255)
    date = models.DateField()
    description = models.TextField()

# Команда
class AddEventCommand:
    def __init__(self, name, date, description):
        self.name = name
        self.date = date
        self.description = description

    def execute(self):
        new_event = Event.objects.create(
            name=self.name,
            date=self.date,
            description=self.description
        )
        return new_event
```

Рисунок 1 – Пример реализации шаблона “Command”

В рамках данного примера модель “Event” представляет мероприятие и содержит три поля: name, date и description. А “AddEventCommand” – это команда, которая инкапсулирует добавление нового мероприятия. Метод execute создает новое мероприятие, используя Django ORM. Далее можно спокойно использовать данный класс в рамках представлений.

Следующий шаблон проектирования – «Итератор» (Iterator). Данный шаблон представляет собой поведенческий паттерн, который обеспечивает способ последовательного доступа к элементам составного объекта, не раскрывая его внутреннего представления. Основная идея заключается в том, чтобы предоставить клиентам единый интерфейс для перебора элементов коллекции, независимо от ее конкретной реализации [3].

Использование паттерна проектирования «Итератор» приносит несколько основных преимуществ:

1. Упрощает классы хранения данных: Применение итератора позволяет достичь более гибкого и расширяемого дизайна, освобождая класс хранения данных от деталей обхода и делая его более универсальным для различных сценариев использования [3].

2. Отделение интерфейса перебора от внутренней реализации: Клиентский код работает с итератором через унифицированный интерфейс, не заботясь о том, как устроена конкретная коллекция. Это позволяет изменять внутреннюю структуру коллекции без изменения кода, использующего итератор.

3. Позволяет одновременно перемещаться по структуре данных в разные стороны: Паттерн предоставляет возможность итерировать коллекцию, как в прямом, так и в обратном направлении. Он расширяет функциональность стандартного итератора, позволяя перемещаться по элементам коллекции как вперед, так и назад.

В Python итераторы представляют собой удобный и интегрированный механизм, но конечно можно реализовать свой собственный итератор, который в рамках класса будет использовать магические методы “__iter__” и “__next__”. Давайте рассмотрим пример реализации данного паттерна в рамках все того же веб-сервиса “Афиша”. Пример реализации можно увидеть на рисунке 2.

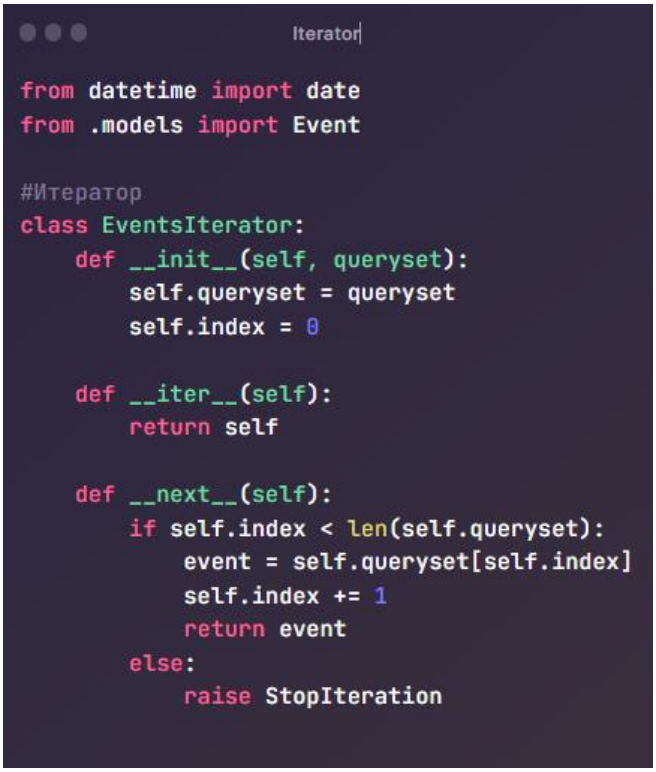

Рисунок 2 – Пример реализации шаблона “Iterator”

В примере реализован “EventsIterator” – это итератор, который позволяет обходить мероприятия в Django QuerySet. Далее можно использовать его в представлениях, например при создании итератора какого-то отфильтрованного QuerySet’а мероприятий, для дальнейшего его использования в приложении.

В заключение хочется отметить, что в данной статье было рассмотрено применение шаблонов проектирования «Команда» и «Итератор» в контексте разработки веб-приложения «Афиша» с использованием языка Python и фреймворка Django. Реализация этих шаблонов в контексте приложения демонстрирует, как можно усовершенствовать структуру приложения, делая его более гибким и легко расширяемым для будущих изменений. При этом обеспечивается поддержание чистоты и читаемости кода, что является фундаментом успешного развития проекта на основе фреймворка Django. Эти шаблоны играют важную роль в улучшении структуры и расширяемости кода, обеспечивая более гибкую и поддерживаемую архитектуру приложения.

Литература

1. Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура. – СПб.: Питер, 2022. – 336 с.: ил. – (Серия «Для профессионалов»).

2. Команда [Электронный ресурс] / Refactoring Guru – 2023. – URL: https://refactoring.guru/ru/design-patterns/command - Дата доступа: 17.12.2023

3. Итератор [Электронный ресурс] / Refactoring Guru – 2023. – URL: https://refactoring.guru/ru/design-patterns/iterator - Дата доступа: 17.12.2023