Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

Технології розроблення програмного забезпечення

«ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY» »

Виконала
студентка групи ІА–22:
Фоменко Альона

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

Тема: ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Мета: Ознайомитися з короткими теоретичними відомостями. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей. Застосування одного з розглянутих шаблонів при реалізації програми.

Теоретичні відомості

Будь-який патерн проектування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проектування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях. Крім того, патерн проектування обов'язково має загальновживане найменування. Правильно сформульований патерн проектування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проектування.

Шаблон «Singleton»

Рис. 1: Шаблон синглтон

Призначення патерну:

«Singleton» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта (звідси і назва «одинак»). Насправді, кількість об'єктів можна задати (тобто не можна створити більш n об'єктів даного класу). Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

Проблема:

Використання одинака виправдано для наступних випадків:

- Може бути не більше N фізичних об'єктів, що відображаються в певних класах;
- Необхідно жорстко контролювати всі операції, що проходять через даний клас.

Одинак вирішує відразу дві проблеми, порушуючи принцип єдиної відповідальності класу.

Рішення:

Перший випадок легко продемонструвати. У кожної програми є певний набір налаштувань, який як правило зберігається в окремий файл (для сучасних комп'ютерних ігор це може бути .ini файл, для .net додатків - .xml файл). Цей файл - єдиний, і тому використання безлічі об'єктів для завантаження/запису даних - нераціональне рішення. Для демонстрації другого випадку, розглянемо наступний приклад. Припустимо, існує дві взаємодіючі системи, між якими встановлено сеанс зв'язку. Накладені обмеження, що по даному сеансу зв'язку дані можуть йти в один момент часу лише в одну сторону. Таким чином, на кожен надісланий запит необхідно дочекатися відповіді, перш ніж відсилати новий запит. Об'єкт «одинак» дозволить не тільки містити рівно один сеанс зв'язку, а й ще реалізувати відповідну логіку перевірок на основі bool операторів про можливість відправки запиту і, можливо, деяку чергу запитів.

Шаблон «Iterator»

Шаблонний ітератор містить:

▪ First() – установка покажчика перебору на перший елемент колекції;

▪ Next() – установка покажчика перебору на наступний елемент колекції;

▪ IsDone – булевське поле, яке встановлюється як true коли покажчик перебору досяг кінця колекції;

▪ CurrentItem – поточний об'єкт колекції.

Призначення:

«Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор - за прохід по колекції. При цьому алгоритм ітератора може змінюватися - при необхідності пройти в зворотньому порядку використовується інший ітератор. Можливо також написання такого ітератора, який проходить список спочатку по парних позиціях (2,4,6-й елементи і т.д.), потім по непарних. Тобто, шаблон ітератор дозволяє реалізовувати різноманітні способи проходження по колекції незалежно від виду і способу представлення даних в колекції.

Шаблон «Proxy»

Призначення:

«Proxy» (Проксі) - об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу. Зазвичай, проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами. Проксі об'єкти використовувалися в більш ранніх версіях інтернет-браузерів, наприклад, для відображення картинки.

В якості іншого прикладу об'єктів заглушок можна привести аспектно-орієнтоване програмування. Наприклад, необхідно додати лог повідомлення при кожному доступі до окремого поля об'єктів певного класу. Є також обмеження, що змінювати програмний код класу не можна. В такому випадку, замість кожного такого об'єкта створюється проксі об'єкт, який при виклику цього методу переадресує виклик реальному об'єкту, але перед цим записує лог повідомлення. Проксі об'єкт повністю повторює визначення об'єкта, який він замінює (тобто всі доступні методи, поля і властивості); проксі об'єкт може також зберігати посилання на реальний об'єкт, в тому випадку, якщо він розширює функціональність, тобто «обгортає» реальний об'єкт.

Шаблон «State»

Призначення:

Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства - бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс.

Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта. Це дозволяє легко додавати в майбутньому і обробляти нові стани, відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

Шаблон «Strategy»

Призначення:

Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує.

Даний шаблон дуже зручний у випадках, коли існують різні «політики» обробки даних. По суті, він дуже схожий на шаблон «State» (Стан), проте використовується в абсолютно інших цілях - незалежно від стану об'єкта відобразити різні можливі поведінки об'єкта (якими досягаються одні й ті самі або схожі цілі).

Хід роботи

Тема 11: Web crawler

Веб-сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

1. Реалізувати не менше 3-х класів відповідно до обраної теми.

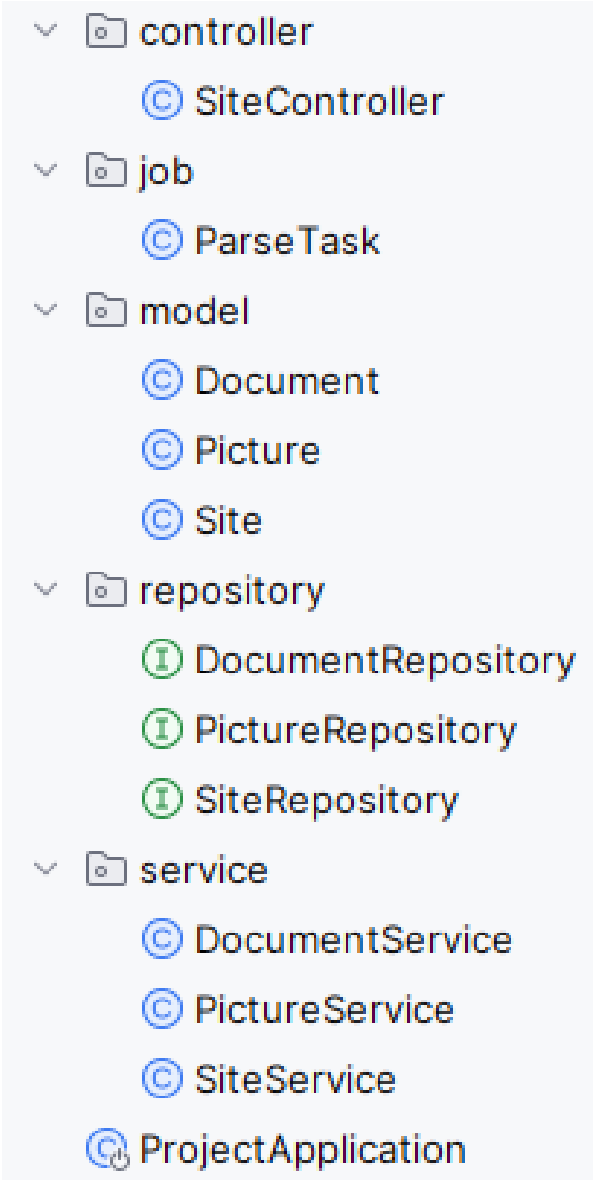

Рис. 2: Структура проекту

Основні компоненти та їх призначення:

1. controller:
    - SiteController — контролер, що обробляє запити, які надходять до системи, та відповідає за взаємодію між клієнтом і сервісами. Він приймає HTTP-запити, обробляє їх і направляє до відповідних сервісів для виконання бізнес-логіки.
2. job:
    - ParseTask — компонент, що реалізує шаблон проєктування Singleton. Це означає, що в системі буде лише один екземпляр цього класу, який використовується для виконання завдань парсингу (наприклад, періодичного завантаження або обробки даних). Singleton забезпечує контрольований доступ до єдиного об'єкта, зручний у випадках, коли потрібно уникнути дублювання ресурсомістких процесів.
3. model:
    - Document, Picture, Site — моделі даних, які описують структуру основних сутностей системи. Вони представляють об'єкти, що зберігаються в базі даних, і містять атрибути, необхідні для роботи з документами, зображеннями та інформацією про сайт.
4. repository:
    - DocumentRepository, PictureRepository, SiteRepository — репозиторії, які виконують операції доступу до бази даних для відповідних моделей. Вони відповідають за зберігання, отримання та маніпуляції даними у сховищі, абстрагуючи ці дії від інших частин системи та забезпечуючи можливість зручної інтеграції з базою даних.
5. service:
    - DocumentService, PictureService, SiteService — сервіси, що реалізують бізнес-логіку для роботи з документами, зображеннями та сайтами. Вони приймають запити від контролерів, обробляють їх за допомогою репозиторіїв і повертають результати. Сервіси служать проміжною ланкою між контролерами та репозиторіями, виконуючи всі необхідні перевірки, обчислення та трансформації даних.

2. Реалізувати один з розглянутих шаблонів за обраною темою.

Цей проект представляє реалізацію crawler, де використовується шаблон проєктування "Singleton", що дозволяє уникнути дублювання обробки даних і забезпечує централізований контроль за завданням краулера.

1. Singlton (ParsTask) - це клас, який реалізує шаблон "Синглтон" і виконує роль основного завдання для парсингу. ParsTask забезпечує єдиний екземпляр у системі, що дозволяє уникнути дублювання обробки даних і забезпечує централізований контроль за завданням краулера.

```
alionafomenko *
@Component
public class ParseTask {

    @Autowired
    PictureService pictureService;
    @Autowired
    DocumentService documentService;

    alionafomenko *
    @Scheduled(fixedDelay = 10000)
    public void parseSite() {

        List<org.example.project.model.Document> documentList = documentService.getAllDocs();
        for (org.example.project.model.Document document : documentList) {
                int level = document.getLevel();
                if (level < 5) {
                    String url = document.getUrl();
                    try {
                        Document doc = Jsoup.connect(url)
                                .userAgent( s: "AlionaCrawler")
                                .timeout( i: 5000)
                                .followRedirects( b: true)
                                .referrer( s: "https://google.com")
                                .get();
                        Elements sites = doc.getAllElements();

                        for (Element el : sites) {
                            String tagName = el.tagName();
                            //adding new urls logic
                            //saving images logic
                            //saving content from site
                        }

                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
        }
    }
}
```

Рис. 3: Реалізація патерну Singleton

Анотація **@Component** вказує на те, що клас ParseTask є компонентом Spring, який автоматично керується контекстом додатку. Завдяки цій анотації Spring створює єдиний екземпляр ParseTask, що реалізує шаблон "Синглтон" і забезпечує доступ до нього в будь-якій частині додатку.

PictureService і DocumentService — це сервіси, які автоматично впроваджуються у ParseTask через анотацію **@Autowired**. DocumentService використовується для отримання списку документів, які містять URL-адреси для парсингу, а PictureService може бути використаний для збереження зображень, знайдених на веб-сторінках.

Метод parseSite() виконується регулярно з інтервалом у 10 секунд, завдяки анотації **@Scheduled**(fixedDelay = 10000). Це означає, що завдання парсингу запускається з фіксованою затримкою у 10 секунд після завершення попереднього виконання. Такий підхід зручний для регулярного збору даних із зазначених веб-сайтів.

Діаграма класів для патерну "Singleton"

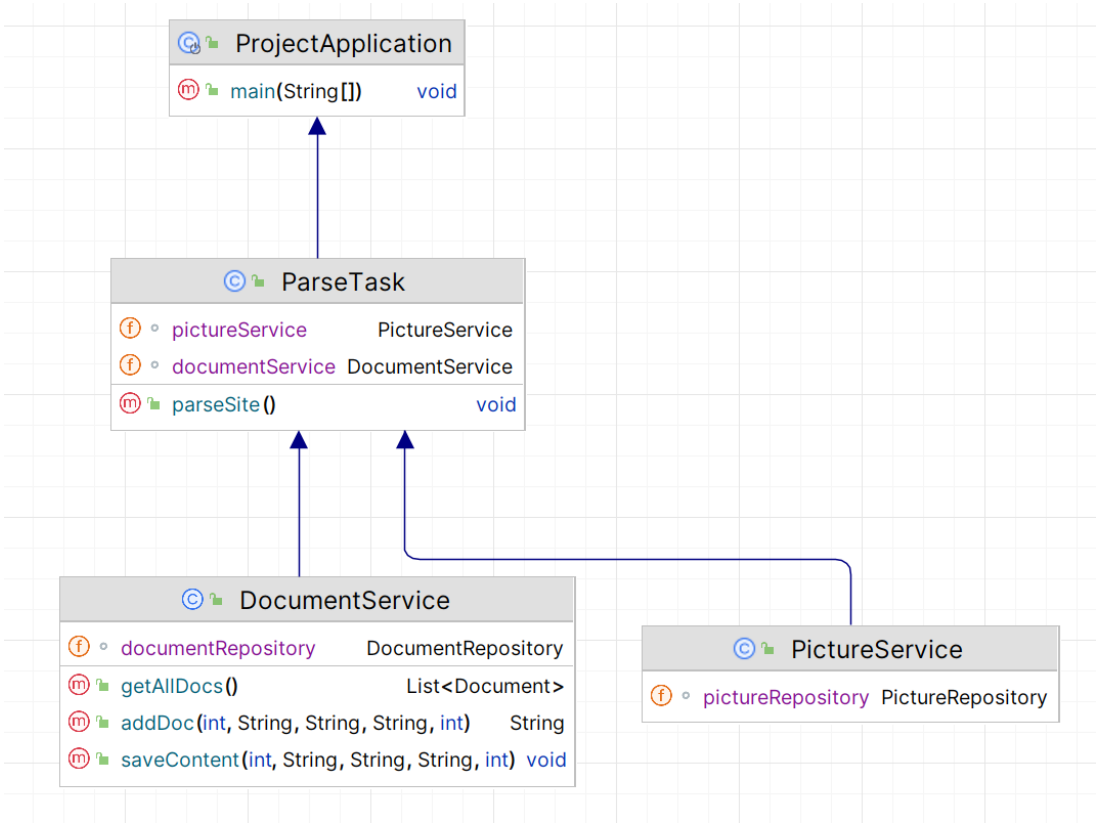

Рис. 4: Діаграма класів патерну Singleton

Проблема яку вирішує патерн "Singleton"

Гарантія єдиного екземпляра:

Замість створення нових об'єктів, "Singleton" зберігає єдиний екземпляр у статичному полі класу. При першому зверненні до класу цей об'єкт ініціалізується (якщо ще не ініціалізований), і всі наступні запити повертають саме цей об'єкт. Це обмежує створення додаткових екземплярів і гарантує, що всі операції проходитимуть через нього.

Наприклад, у ParseTask це означає, що вся логіка краулінгу зосереджена в одному місці. Така структура дає можливість встановити контроль над потоками, що запускають завдання, і забезпечити стабільність роботи програми.

Централізоване управління ресурсами:

Завдяки єдиній точці доступу, "Singleton" дозволяє централізовано керувати ресурсами, такими як бази даних, мережеві з'єднання або складні операції, які вимагають синхронізації. Це полегшує моніторинг і управління ресурсами, що є критично важливим для продуктивності.

У ParseTask, наприклад, можна керувати інтервалами роботи завдання, моніторити його стан і забезпечувати належне використання мережевих з'єднань для збирання даних. У разі потреби можна також налаштувати ліміти на кількість запитів до одного сайту або встановити обмеження на завантаження ресурсів, що підвищує стабільність системи.

Переваги використання патерну «Singleton»

Однак слід зазначити, що в даний час «одинака» багато хто вважає т.зв. «анти-шаблоном», тобто поганою практикою проектування. Це пов'язано з тим, що «одинаки» представляють собою глобальні дані (як глобальна змінна), що мають стан. Стан глобальних об'єктів важко відслідковувати і підтримувати коректно; також глобальні об'єкти важко тестуються і вносять складність в програмний код.

При цьому реалізація контролю доступу можлива за допомогою статичних змінних, замикань, мютексов та інших спеціальних структур.

+ Гарантує наявність єдиного екземпляра класу.

+ Надає до нього глобальну точку доступу.

Висновок: У даній лабораторній роботі я реалізувала патерн проєктування Singleton для системи веб-краулера, що дозволило створити єдиний екземпляр класу, відповідального за парсинг веб-сторінок. Це забезпечило централізований контроль за процесом збору даних і уникнення дублювання запитів до тих самих ресурсів. Завдяки патерну Singleton логіка обробки URL-адрес була зосереджена в одному класі, що значно спростило управління і масштабування системи. Переваги використання Singleton включають обмеження створення зайвих об'єктів, зручність управління ресурсами та підвищену стабільність системи, що покращує її підтримку та розширення в майбутньому.