Розробка пошукового робота з можливістю гнучкої конфігурації

Глибовець Андрій Миколайович
доцент
Національний університет «Києво-Могилянська академія»
Україна, Київ

Проектування та побудова пошукового робота - досить нетривіальна задача. Актуальність даної теми полягає в тому, що не зважаючи на важливість пошукових роботів для ефективного розвитку пошукових та інформаційних систем загалом, на сьогодні усе ж дана проблема не є повністю вирішеною, існує досить небагато відкритих реалізацій пошукових роботів, які б можна було просто, швидко та гнучко налаштувати для своїх цілей та використовувати у своїх проектах. В роботі було запропоновано один з підходів до побудови пошукового робота з можливістю конфігурації.

Ключові слова: інформаційно пошукові системи, пошуковий робот

ВСТУП

На сьогодні неможливо уявити жодну інформаційну систему без можливостей пошуку, аналізу та збереження інформації. Будь-яка пошукова система повинна вміти проводити індексацію та аналіз (наприклад, веб- сторінок) для того аби надавати релевантну інформацію у відповідь на пошукові запити користувача в майбутньому. Для цих цілей широко використовують пошукових роботів, які проводять аналіз та індексацію веб- сторінок з метою пришвидшення роботи системи.

АНАЛІЗ ПРОБЛЕМ РОЗРОБКИ ПОШУКОВОГО РОБОТА

Пошуковий робот - це програма, яка аналізує так індексує веб-сторінки для подальшого використання цього індексу пошуковою системою та побудови веб-кешу [1]. Робот починає свою роботу із заданими посиланнями на веб-сторінки, при проведенні аналізу кожної сторінки робот виокремлює посилання на ресурси та додає їх у свою внутрішню чергу для подальшої обробки. Варто додати, що деякі реалізації пошукових збирачів вміють аналізувати вміст сторінки і на основі аналізу, ефективно відшукувати лише необхідну інформацію. Кожну успішно проаналізовану сторінку робот повертає для подальшого її аналізу клієнтом. Робот продовжує аналіз сторінок поки черга на обробку не стане порожньою.

Першою проблемою, яка виникає при реалізації такого алгоритму роботи є проблема зберігання проаналізованих сторінок, оскільки клієнти можуть мати досить обмежені ресурси.

Наступною проблемою є проблема доступу до сервера, на якому зберігаються веб-сторінки. Так, наприклад, погано спроектований або налаштований робот може спричинити відмову в обслуговувані інших клієнтів, якщо він буде занадто часто доступатися до веб-сторінок і у такий спосіб перенавантажить сервер.

Іноді може трапитися ситуація, що за посиланням вже немає вказаного ресурс або веб-сервер повертає помилку при обробці запиту. Із цих ситуацій випливають підвищені вимоги до надійності та стабільності у роботі пошукового робота, він має продовжувати аналіз електронного ресурсу навіть за наявності вищезгаданих помилок та відмов в обслуговуванні, по- можливості, інформуючи при цьому свого клієнта.

Аналіз навіть невеликого електронного ресурсу потребує часу, тим більше це стосується великих веб-порталів. За таких обставин можна стикнутися з проблемою, коли під час аналізу робот має повертатися до вже проаналізованих сторінок із метою отримання актуальної інформації. Із ростом розміру порталу та складністю алгоритму робота ситуація стає ще більш складною.

Таким чином, можна зробити висновок, що розробка пошукового робота – досить складна та нетривіальна задача із великою кількістю проблем [2].

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПОШУКОВОГО РОБОТА

При проектуванні та розробці було приділено багато уваги можливості простого налаштування та вдосконалення робота в майбутньому. Основні проблеми, які необхідно було вирішити при проектуванні - це мати можливість налаштовувати робота під конкретний веб-ресурс, не змінюючи при цьому вихідний код системи та зробити компоненти системи якомога менше зв’язаними, а отже більш зручними та придатними для подальшого вдосконалення.

Для розв’язання проблеми налаштування було прийнято рішення розробити власний XML формат, який би у зручний спосіб давав змогу налаштувати робота. Для полегшення процесу модифікації системи було вирішено додати системний сервіс, який би на основі будь-якого зручного для користувача формату файла налаштувань (XML, JSON, YAML та інших) генерував би “зрозумілий” системі екземпляр класу налаштувань. При такому підході стає можливим легко замінювати один тип файлу налаштувань іншим. UML діаграму такого модуля представлено рисунком 1

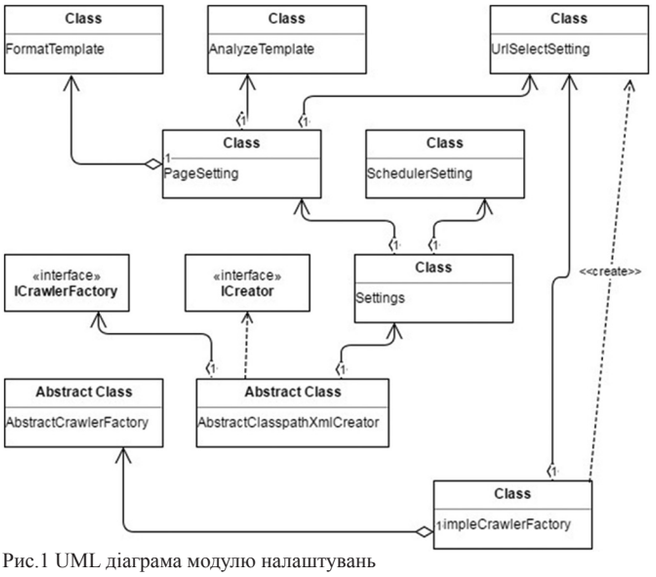

Рис.1 UML діаграма модулю налаштувань

Основним класом у даній підсистемі є клас Settings, який містить основні налаштування системи: аналізу й форматування веб-сторінок і також налаштування планувальника задач, які представлені класами PageSetting та SchedulerSetting відповідно.

Налаштування планувальника задач повинні надавати змогу регулювати кількість потоків, що може використовувати робот для поточної переіндексації та обробки сторінок. Окрім того, для зменшення навантаження на веб-сервер, варто ввести можливість регульованої затримки між послідовними обробками веб-сторінок.

Фактичне створення екземплярів пошукового робота беруть на себе класи, що реалізовують інтерфейс ICreator. Класи, що наслідують інтерфейс ICrawlerFactory також відповідають за створення екземпляру робота, але вже на основі готових екземплярів типу Settings.

Для того аби мати можливість підтримувати новий формат налаштувань достатньо реалізувати інтерфейс ICreator, при цьому можна використати існуючу реалізацію інтерфейсу ICrawlerFactory або замінити її на іншу.

Розробка формату налаштувань

XML досить часто використовують для написання файлів конфігурації через те, що його досить легко перевірити на коректність за допомогою XSD схем, до того ж XML досить близький до людської мови, а тому досить простий для розуміння людиною.

Якщо проаналізувати вищенаведені вимоги, які так чи інакше стосуються налаштувань пошукового робота, то можна зробити висновок, що конфігурація пошукового робота повинна мати такі можливості:

- Вказувати початковий список посилань на веб-сторінки
- Мати базові налаштування планувальника задач:
  - Можливість вказувати затримку між поточною обробкою сторінок
  - Можливість вказати кількість потоків, що їх може використовувати робот
- Мати можливість створювати шаблон сторінки:
  - Вказувати унікальний ідентифікатор сторінки
  - Вказувати налаштування аналізатора
  - Вказувати налаштування форматування

Приклад налаштувань:

```
<?xml version=»1.0» encoding=»UTF-8» ?>
<crawler>
  <urls>
<url>http://nz.ukma.edu.ua/index.php?option=com_content&
amp;task=section&amp;id=10&amp;Itemid=47</url>
  </urls>
  <scheduler-params>
    <startup-delay>0</startup-delay>
    <index-delay>60000</index-delay>
    <threads>2</threads>
  </scheduler-params>
  <pages>
    <page id=»1»>
      <analyze-params>
        <analyze weight=»10»>body > table > tbody > tr >
td > table:nth-child(5) > tbody > tr > td:nth-child(2) > table >
tbody > tr:nth-child(3) > td > ul > li > a</analyze>
      </analyze-params>
      <url-params>
        <extract attr=»href»
                selector=»body > table > tbody > tr > td >
table:nth-child(5) > tbody > tr > td:nth-child(2) > table > tbody
> tr:nth-child(3) > td > ul > li > a»/>
      </url-params>
    </page>
    <page id=»2»>
      <analyze-params>
        <analyze weight=»70»>body > table > tbody > tr >
td > table:nth-child(5) > tbody > tr > td:nth-child(2) > table >
tbody > tr:nth-child(2) > td > form > table > tbody > tr > td >
a</analyze>
      </analyze-params>
      <url-params>
        <extract attr=»href»
                selector=»body > table > tbody > tr > td >
table:nth-child(5) > tbody > tr > td:nth-child(2) > table > tbody
> tr:nth-child(2) > td > form > table > tbody > tr > td > a»/>
      </url-params>
    </page>
  </pages>
</crawler>
```

Розроблений формат налаштувань дозволяє зручно та швидко створювати шаблони сторінок із метою подальшого аналізу. Так, можливо вказати унікальний ідентифікатор для кожного шаблону та мінімальне порогове значення важливості, при цьому кожен шаблон може містити необмежену кількість правил аналізу, форматування і вилучення посилань зі сторінки (xml-елементи analyze та extract відповідно). Кожне таке правило є CSS-селектором, що дає змогу легко реалізувати роботу з DOM-структурою. Для валідації XML документу було створено XSD схему

ОПИС ВИКОРИСТАНИХ ЗАСОБІВ

Для розробки пошукового робота було вирішено використовувати мінімальну кількість сторонніх бібліотек задля зменшення залежностей від них й, відповідно за необхідності, полегшення процесу переходу на інші; також одним із критеріїв вибору інструментарію були простота у використанні, стабільність та поширеність.

Пошукового робота було розроблено із використанням б'єктно-орієнтованої мови програмування Java версії 1.8. У якості IDE було обрано Intellij Idea Community Edition 2016. Перелік використаних бібліотек включає в себе:

- Lombok версії 1.16.12, бібліотека для автоматичної генерації коду, допомагає усувати шаблонний код
- H2 database connector версії 1.3.170, з'єднання з базою даних H2
- JSoup версії 1.10.1, бібліотека для аналізу та модифікації DOM, побудови CSS селекторів
- Javax validation annotations версії 1.1.0, анотації для статичного аналізу коду засобами Intellij Idea.

ВИСНОВКИ

У ході виконання роботи було виявлено та проаналізовано основні проблеми, що виникають при розробці пошукових роботів: можливість створення великого навантаження на аналізований веб-ресурс, обмеженість у ресурсах як аналізатора, так клієнта, високі вимоги до надійності та тривалість процесу індексації.

На основі проаналізованої інформації нами була спроектована UML діаграма класів, на основі якої вже було реалізовано власне сам пошуковий робот. У ході проектування були наведені аргументи на рахунок вибору того чи іншого підходу розробки. Наступним кроком було створення XML формат налаштувань який задовольняв висунутим вимогам та дозволяв швидко конфігурувати робота; для валідації XML файлу налаштувань була розроблена окрема XSD схема.

Для пришвидшення процесу розробки використовувались сторонні бібліотеки. Ключовим фактором при виборі того чи іншого інструменту були простота у використання та надійність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Manning C. Introduction to Information Retrieval / C. Manning, P. Raghavan, H. Schütze., 2008. – 482 с.

2. Глибовець А.М., Шабінський А.С. Один підхід до побудови інтелектуальної пошукової системи // Наукові записки НАУКМА. Комп'ютерні науки. —том 112. — 2010. - с. 26-30. (4 ст.)