ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП

ЗБІРНИК МАТЕРІАЛІВ

Міжнародної науково-практичної Інтернет-конференції

20-21 листопада 2023 р.

УДК 004
ББК 32.97
Е50

Рекомендовано до видання Вченою радою КЗВО «Вінницька академія безперервної освіти» (протокол № 8 від 20.11.2023 р.)

Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – 336 с.

ISBN 978-617-7422-23-4

Збірник містить матеріали Міжнародної науково-практичної Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ. Матеріали збірника подано у авторській редакції. Автори опублікованих матеріалів несуть повну відповідальність за підбір, точність наведених фактів, цитат, статистичних даних, власних імен та інших відомостей, Матеріали відтворюються зі збереженням змісту, орфографії та синтаксису текстів, наданих авторами.

УДК 004
ISBN 978-617-7422-23-4

© КЗВО «Вінницька академія безперервної освіти», 2023
© Вид-во Суми, НІКО, 2023

СТАРІКОВ І.Р., ТРОФИМЕНКО О.Г.
НУ «Одеська юридична академія»

ЗАСТОСУВАННЯ СТРУКТУР ДАНИХ У BACK-END ЗАСОБАМИ JAVA

Анотація: Розглянуто на охарактеризовано структури даних, які є в мові програмування Java і можуть використовуватися для розробки back-end. Акцентовано увагу на важливості розуміння розробником back-end та вміння використання ефективних структур даних при створенні серверної сторони програмного забезпечення.

Ключові слова: back-end, структури даних, Java, List, Map, хешування.

Back-end є популярною сферою, проте висока конкуренція та постійне підвищення вимог до вебпродуктів формують зростання вимог до компетентностей фахівців цієї галузі розробки програмного забезпечення. По суті, розробник back-end відповідає за створення, розвиток та підтримку серверної сторони програмного забезпечення. Саме він відіграє ключову роль у забезпеченні надійності та високої продуктивності програмних рішень, оскільки керує створенням та обслуговуванням інфраструктури, яка відповідає на запити клієнтів. Його завданнями є робота з базами даних з ефективним обміном інформацією між клієнтом та сервером, оброблення запитів від інтерфейсу користувача або front-end, а також забезпечення безпеки, продуктивності і стабільності програмних систем. Тому для роботи бекенд-розробником наразі недостатньо вивчення і знання деяких фреймворків, мови SQL та основ керування базами даних. До інструментарію розробника back-end входять різноманітні структури та хешування, сортування і пошуку.

У найбільш поширеній для back-end мові програмування Java використання структур List, Stack, Queue, Set, Map має свою специфіку. Так, у Java масиви для значень-посилань реалізовані структурами List, Set та Queue, з яких найчастіше використовується List. Приклад, коли дані від стороннього сервісу після отримання результату зберігаються в List-списку:

```
List<IContractBuilderVariable> variables =
  contractBuilderClauseService.fetchVariablesByDomainAndClauseContent (
        domain, clauseContent, locale);
```

До речі, такі структури є параметризованими, що прискорює їхню роботу і робіть її більш надійною. Використання інтерфейсів List< >, замість реалізацій ArrayList< > чи то LinkedList< >, називається "програмуванням на рівні інтерфейсу" або "програмуванням на рівні абстракцій", що вважається гарною практикою програмування з різних причин:

- *гнучкість та абстракція*: використання інтерфейсів створює абстракцію для конкретної реалізації і тим самим дозволяє легко змінювати реалізацію без змінення коду, що використовує цей інтерфейс;
- *розширюваність*: інтерфейси дозволяють розширювати функціональність, не порушуючи наявного коду. Якщо надалі знадобиться додавати нову реалізацію List, то все, що використовує інтерфейс, залишиться незмінним;
- *тестованість*: використання інтерфейсів допомагає створювати макети (mocks) та модульні тести для коду, чим полегшує тестування, оскільки дозволяє імітувати різні реалізації List у тестах;
- *принципи SOLID* (наприклад, принципи інверсії залежностей та розподілу інтерфейсу) підтримують використання інтерфейсів та абстракцій для зменшення зв'язності й збільшення гнучкості коду. Тим самим використання інтерфейсів, замість конкретних реалізацій, є гарною практикою, яка сприяє більш гнучкому і підтримуваному коду.

Структури даних List, Stack і Queue є базовими структурами даних у програмуванні і використовуються для різних задач. При цьому List використовується частіше, бо є більш загальним і універсальним інструментом. Перевагами List є: можливість доступу до будь-якого елемента через індекс, можливість додавати, видаляти та модифікувати елементи. При цьому суттєвим є те, що подання List може бути різним, наприклад, ArrayList використовує масив, тоді як LinkedList використовує зв'язний список.

Структура даних Stack керується принципом LIFO: Last-In-First-Out, останній елемент, доданий у стек, вийде з нього першим. Базовими операціями стека є: push (додати на вершину) і pop (вилучити з вершини). Stack використовується для розв'язання задач, для яких важливим є порядок видалення елементів, як-от рекурсія.

Структура даних Queue послуговується принципом FIFO: First-In-First-Out, перший, доданий у чергу елемент вийде першим. Queue використовується для розв'язання задач, для яких важливим є відповідний порядок оброблення елементів. Stack і Queue використовуються для конкретних сценаріїв й обмежуються в термінах доступу та операцій [1].

Окрім розглянутих базових структур, у Java є структура Map, яку організовано за принципом «ключ-значення»:

```
public Map<String, Object> getSupplierAdditionalParametersMap() {
   Map<String, Object> additionalParameters = new HashMap<>();
   additionalParameters.put(PARAM_CONTRACT_UUID, getContractUUID());
   return additionalParameters;
}
```

Тут getSupplierAdditionalParametersMap виконує збір параметрів, які потім можуть передаватися через інтерфейс Map. Поширено відома реалізація HashMap використовує хешкод (адресу пам'яті) для швидкого доступу до значення. HashMap використовують для передачі даних по запиту від сервера до клієнта, оскільки ключ завжди є унікальним. Крім того, є структура HashTable, яка використовується вкрай рідко, оскільки є синхронізованою та неефективною щодо часу виконання операцій над даними.

Окрім HashMap та HashTable, у Java є ще структура HashSet, яка теж використовує хешування для зберігання даних і зберігає пари «ключ-значення». Однак, на відміну від HashMap, HashSet зберігає лише унікальні об'єкти. Якщо в HashMap унікальними повинні бути лише ключі і при цьому значення можуть однаковими бути, то в HashSet всі об'єкти повинні бути унікальними. Саме тому HashMap використовують, коли треба швидко отримати доступ до значення через ключ, а HashSet – коли потрібно швидко перевірити наявність конкретного елемента.

Керування базами даних у back-end є критично важливим аспектом розробки серверної частини вебсайту і вимагає досвіду співпраці зі структурами бази даних. Backend-розробник прагне забезпечити ефективне та масштабоване функціонування серверної частини, проводячи оптимізацію коду, застосування ефективних структур даних та налаштувань сервера. На вибір тих чи інших структур даних для конкретної задачі впливає кількість даних, частота операцій додавання, видалення і пошуку, а також необхідність сортування або гарантії унікальності елементів. Правильний їх вибір може значно покращити продуктивність та ефективність програмного забезпечення, тому важливо глибоко розуміти всі наявні опції та вміти їх використовувати.

Список використаних джерел

1. Колекції в Java. Частина 2: HashSet, HashMap та інші. URL: https://mate.academy/blog/java-development/java-collections-2/

ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП:

Збірник матеріалів
Міжнародної науково-практичної Інтернет-конференції
20-21 листопада 2023 р.

Редактор С.А.Пойда, М.С. Ніколаєнко
Комп'ютерне верстання С.А.Пойда, М.С. Ніколаєнко

Підписано до друку 15.11.2023 Гарнітура Times New Roman
Формат 60х84/16 Папір офсетний
Друк цифровий Ум. друк. арк. 19,4
Тираж 300 пр. Зам. № 2/23

Видавництво НІКО
м.Суми, вул.Харківська, 54
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія СМв № 044
від 15.10.2012
E-mail: ms.niko@i.ua
Телефон для замовлень: +38(066) 270-64-68