Івано-Франківський національний технічний

університет нафти і газу

Кафедра

інженерії програмного забезпечення

Лабораторна робота №5

Бібліотеки Collections Generics та LINQ в C#

Виконав

Ст. гр. ІП-22-1

Хімій Денис

Перевірила

Піх М.М.

Івано-Франківськ

2024

Мета: проаналізувати застосування та переваги стандартних бібліотек System Collections Generics та LINQ, які є аналогом System Template Library в C++. Продемонструвати їх застосування у коді проєкту.

Хід виконання роботи

Collections Generics

Це простір імен у .NET Framework, який надає розробникам потужні інструменти для створення узагальнених колекцій, що забезпечують безпеку типізації, продуктивність та гнучкість. Використання узагальнених колекцій дозволяє працювати з різними типами даних, уникаючи помилок, пов'язаних із невідповідністю типів, і оптимізуючи операції доступу та обробки даних. Ці колекції значно спрощують керування великими обсягами даних, пропонуючи широкий вибір структур даних, таких як списки, словники, черги, стеки та множини, що робить їх незамінними для потреб сучасного програмування.

Особливості

Безпека типізації:

Узагальнені колекції гарантують, що всі елементи колекції мають однаковий тип, що запобігає помилкам, пов'язаним із невідповідністю типів даних.

Продуктивність:

Завдяки тому, що узагальнені колекції працюють з конкретними типами даних, відпадає потреба в перетворенні типів (boxing/unboxing), що підвищує продуктивність додатків.

Гнучкість:

Узагальнені колекції можуть працювати з будь-яким типом даних, що робить їх універсальними та зручними для використання в різних сценаріях.

Переваги

Зручність

Узагальнені колекції мають зрозумілий інтерфейс для додавання, видалення та доступу до елементів.

Широкий вибір колекцій (структур даних)

У просторі імен System.Collections.Generic доступні різні типи колекцій, такі як списки, словники, черги, стеки та множини, що дозволяє вибрати найкращий варіант для конкретної задачі.

Масштабованість

Узагальнені колекції можуть динамічно змінювати свій розмір, що робить їх придатними для роботи з великими обсягами даних.

Основні колекції

List<T>

Динамічний масив, що дозволяє додавати, видаляти та змінювати елементи.

Операції

- Add: Додає новий елемент в кінець списку.
- Remove: Видаляє перший знайдений елемент зі списку.
- Insert: Вставляє елемент у вказану позицію у списку.
- Count: Повертає кількість елементів у списку.
- Clear: Очищає весь список, видаляючи всі елементи

Dictionary<TKey, TValue>

Колекція пар "ключ-значення", що забезпечує швидкий доступ до значень за ключами.

Операції

- Add: Додає нову пару ключ-значення у словник.
- Remove: Видаляє пару зі словника за вказаним ключем.

- ContainsKey: Перевіряє наявність ключа у словнику.
- TryGetValue: Отримує значення за ключем, якщо він присутній.
- Count: Повертає кількість пар "ключ-значення" у словнику.

Queue<T>

Черга, що працює за принципом "перший прийшов - перший пішов" (FIFO).

Операції

- Enqueue: Додає елемент в кінець черги.
- Dequeue: Видаляє та повертає перший елемент з черги.
- Peek: Повертає перший елемент без видалення з черги.
- Count: Повертає кількість елементів у черзі

Stack<T>

Стек, що працює за принципом "останній прийшов - перший пішов" (LIFO).

Операції

- Push: Додає елемент на вершину стеку.
- Pop: Видаляє та повертає елемент з вершини стеку.
- Peek: Повертає елемент на вершині без видалення зі стеку.
- Count: Повертає кількість елементів у стеці.

HashSet<T>

Колекція унікальних елементів, що забезпечує швидкі операції додавання, видалення та пошуку.

Операції

- Add: Додає новий унікальний елемент у множину.
- Remove: Видаляє елемент із множини.
- Contains: Перевіряє чи включений елемент у множину.
- Count: Повертає кількість елементів у множині.

Ітератори

Ітератори у мові програмування C# є потужним механізмом для ітерації або перебору елементів в колекціях чи власних структурах даних. Вони дозволяють реалізувати ліниву оцінку (lazy evaluation), що означає, що елементи обробляються лише в момент їх запиту. Це забезпечує оптимальне використання ресурсів та покращує продуктивність програм. Ітератори в C# реалізовані за допомогою ключового слова yield, яке дозволяє методу повертати послідовності значень, що можуть бути ітеровані.

При використанні ітераторів можна створювати власні ітератори для будь-яких колекцій або даних, навіть для нестандартних структур, які не підтримують стандартні інтерфейси. Це дозволяє забезпечити єдиний інтерфейс для перебору даних, незалежно від їхнього джерела. Ітератори в C# також підтримують узагальнення (generics), що робить їх універсальними і гнучкими для різноманітних використань у програмуванні.

LINQ

Language Integrated Query – компонент Microsoft .NET Framework, який додає нативні можливості виконання запитів даних до мов, що входять у .NET, включаючи C#.

LINQ розширює можливості мови, додаючи до неї вирази запитів, що є схожими на твердження SQL та можуть бути використані для зручного отримання та обробки даних масивів, XML документів, реляційних баз даних та сторонніх джерел. LINQ також визначає набір імен методів (що називаються стандартними операторами запитів, або стандартними операторами послідовностей), а також правила перекладу, що має використовувати компілятор для перекладу текучих виразів у звичайні, використовуючи їх назву, лямбда-вирази та анонімні типи.

Фільтрація – Where

Використовується для відбору елементів з колекції, що задовольняють певну умову.

Сортування – Sort

Використовується для сортування елементів колекції за певними критеріями.

Проєкція – Select

Використовується для створення нових об'єктів або трансформації даних.

Групування – GroupBy

Використовується для групування елементів за певними ключами.

Об’єднання – Join

Використовується для об'єднання двох або більше колекцій за певними умовами.

Агрегація – Aggregate, Sum, Average, Min, Max

Використовується для обчислення суми, середнього значення, мінімального або максимального значення елементів колекції.

Застосування Generics та LINQ

На прикладі кількох запитів із проєкту можна наглядно показати як поєднання бібліотек System.Colletion.Generics та System.LINQ дозволяє здійснювати CRUD-операції безпосередньо в C#-коді проєкту на противагу використанню для цього мови SQL.

Запит 1

Перелік видів виробів окремої категорії і в цілому, що збираються зазначеним цехом, підприємством.

PostgreSQL

```
select * from (select id, name as category from product_categories) pc
    join products op on pc.id = op.category
    left join cars c on op.id = c.id
    left join bikes on op.id = bikes.id
    left join buses on op.id = buses.id
    left join trucks tk on op.id = tk.id
    left join tractors tr on op.id = tr.id
    left join excavator e on op.id = e.id
where op.id in (
```

```
    select distinct p.id from products p
        join brigades_specialization bs on p.id = bs.product
        join brigades b on b.id = bs.id
        join sites s on s.id = b.site
        join workshops w on w.id = s.workshop;
```

C#

```
public List<Workshop> Workshops { get; protected set; }
public HashSet<Product> Products { get; protected set; }


public HashSet<Product> Select(Category category = default, int workshop = -1)
{
       return
              (workshop == -1 ? Products : Workshops[workshop].Products).
              Where(p => category == default || p.Category == category).ToHashSet();
}
```

Запит 2

Дані про кадровий склад цеху, підприємства в цілому і по зазначеним категоріям інженерно-технічного персоналу і робітників.

PostgreSQL

```
select ec.name, count(e.id) from engineers e
    left join engineer_categories ec on e.category = ec.id
    left join sites_masters sm on e.id = sm.id
    left join sites s on sm.site = s.id or e.id = s.chief
where
     s.workshop = 1 and s.id = 3
group by ec.name
union all
select lc.name, count(l.id) from laborers l
    left join laborer_categories lc on l.category = lc.id
    left join brigades_staff bs on l.id = bs.id
    left join brigades b on bs.brigade = b.id or l.id = b.foreman
    left join sites s on b.site = s.id
where
    s.id = 2
group by lc.name;
```

C#

```
public List<Site> Sites { get; private set; }
public List<Workshop> Workshops { get; protected set; }

public List<Person> Select
       (int workshop = -1,
       EmployeeType employeeType = EmployeeType.All,
       EngineerCategory engineer = default,
       LaborerCategory laborer = default)
{
```

```
	return
		Workshops.Where
		(
			w => workshop == -1 ||
			Workshops.IndexOf(w) == workshop
		)
		.SelectMany(w => w.Sites).SelectMany(
			s => s.Brigades.SelectMany(b =>
			b.Laborers.Where
		(
			l => employeeType == EmployeeType.All ||
			(employeeType == EmployeeType.Laborer && l.Category == laborer)
		)
		.Select(l => l as Person)).
		Concat(s.Engineers.Where
		(
			e => employeeType == EmployeeType.All ||
			(employeeType == EmployeeType.Engineer && e.Category == engineer)
		).
		Select(e => e as Person))).ToList();
}
```

Запит 3

Склад бригад зазначеної ділянки, цеху.

PostgreSQL

```
select
    first_name, last_name, age, experience, lc.name as category, speed, detect_rate,
material_saving,  accuracy, processing_level, b.name as brigade, b.id as brigade_id
from staff s
    join laborers l on s.id = l.id
    left join laborer_categories lc on l.category = lc.id
    left join installers i on l.id = i.id
    left join welders w on l.id = w.id
    left join turners t3 on l.id = t3.id
    left join millers m1 on l.id = m1.id
    left join grinders g on l.id = g.id
    left join brigades_staff bs on l.id = bs.id
    left join brigades b on l.id = b.foreman or bs.brigade = b.id
    left join sites s1 on s1.id = b.site
where
    s1 = 3 and
    s1.worksop = 1;
```

C#

```
public List<Site> Sites { get; private set; }
public List<Workshop> Workshops { get; protected set; }

public List<Laborer> Select(int workshop = -1, int site = -1)
{
	return
		Workshops.Where
```

```
                (
                        w => workshop == -1 ||
                        Workshops.IndexOf(w) == workshop
                )
                .SelectMany(w => w.Sites.Where
                (
                        s => site == -1 ||
                        Sites.IndexOf(s) == site
                )
                .SelectMany(s => s.Brigades)
                .SelectMany(b => b.Laborers)).ToList();
}
```

Висновок

На цій лабораторній роботі було проаналізовано застосування та переваги стандартних бібліотек System Collections Generics та LINQ. Продемонстровано їх застосування у коді проєкту.