Платформа .NET версии 3.5 представила новую технологию работы с коллекциями объектов. Данная технология носит название *Language Integrated Query* (LINQ). Технически, LINQ – это набор классов, содержащих типичные методы работы с коллекциями: поиск данных, сортировка, фильтрация. Для достижения универсальности и легкости при использовании, большинство методов объявлены как методы расширения интерфейса IEnumerable<T>. Многие методы реализованы с применением итераторов и поддерживают отложенные вычисления.

Принято разделять технологию LINQ по типу обрабатываемой информации. LINQ to Objects – это библиотеки для обработки коллекций объектов в памяти; LINQ to SQL – библиотеки для работы с базами данных; LINQ to XML предназначена для обработки XML-информации. В данном параграфе акцент сделан на библиотеке LINQ to Objects.

Ядром технологии LINQ to Objects является статический класс Enumerable, размещенный в пространстве имен System.Linq[[1]](#footnote-1). Этот класс содержит набор методов расширения интерфейса IEnumerable<T>, которые в дальнейшем будут называться *операторами LINQ*.

LINQ предоставляет кучу операторов для работы с объектами. В их числе:

**Оператор условия Where.** Оператор производит фильтрацию коллекции, основываясь на параметре-предикате. Поддерживает отложенные вычисления.

var result = lst.Where(x => x < 0);

**Операторы проекций.** *Анонимные типы* позволяют создавать новый тип, не декларируя его заранее, а описывая непосредственно при создании переменной. При обработке коллекций тип элементов результата может отличаться от типа элементов исходной коллекции.

При объявлении анонимного типа используется ключевое слово new, затем в фигурных скобках указываются через запятую пары вида «имя поля = значение поля». Тип поля не указывается, а выводится из начального значения.

var anon = new { a = 3, b = 4.81, c = "string data" };

Анонимный тип следует рассматривать как класс, состоящий из полей только для чтения. Кроме полей, других элементов анонимный тип содержать не может. Два анонимных типа считаются эквивалентными, если у них полностью (вплоть до порядка) совпадают поля.

Операторы проекций применяются для выборки информации, при этом они могут изменять тип элементов итоговой коллекции. Основным оператором проекции является Select().

Оператор SelectMany() может применяться в том случае, если результатом проекции является набор данных. В этом случае оператор соединяет все элементы набора в одну коллекцию.

var r2 = gr.Select(s => new { Name = s.Name, Age = s.Age});

var r3 = gr.SelectMany(s => s.Marks);

**Операторы упорядочивания.**

Данные операторы выполняют сортировку коллекций. Операторы OrderBy() и OrderByDescending() выполняют сортировку по возрастанию или убыванию соответственно.

**Операторы группировки.** Операторы группировки позволяют объединить некоторые элементы коллекции, если у этих элементов одинаковое значение определенного поля-ключа GroupBy().

**Операторы соединения.** Операторы соединения применяются, когда требуется соединить две коллекции, элементы которых имеют общие атрибуты. Основным оператором соединения является оператор Join().

var res = gr.Join(cit, s => 2008 - s.Age, c => c.YearOfBirth,

(s, c) => new {s.Name, Number = c.PassportNumber});

Также в LINQ имеется оператор GroupJoin(), который работает как операторы языка SQL LEFT OUTER JOIN или RIGHT OUTER JOIN.

|  |  |
| --- | --- |
| **Оператор** | **Описание** |
| **Объединение** |  |
| Aggregate | Применяет к последовательности пользовательский метод. |
| Average | Вычисляет среднее для числовой последовательности. |
| Count | Возвращает количество элементов в последовательности (целочисленное значение). |
| LongCount | Возвращает количество элементов в последовательности (значение в диапазоне LongInt). |
| Min | Возвращает наименьшее значение для числовой последовательности. |
| Max | Возвращает наибольшее значение для числовой последовательности. |
| Sum | Складывает члены числовой последовательности. |
| **Конкатенация** |  |
| Concat | Соединяет две последовательности в одну. |
| **Преобразование** |  |
| Cast | Преобразует элементы последовательности в элемены указанного типа. |
| OfType | Выбирает из элементов последовательности элемены указанного типа. |
| ToArray | Возвращает массив из элементов последовательности. |
| ToDictionary | Возвращает словарь из элементов последовательности. |
| ToList | Возвращает список из элементов последовательности. |
| ToLookup | Возвращает результаты поиска по последовательности. |
| ToSequence | Возвращает последовательность IEnumerable. |
| **Элемент** |  |
| DefaultIfEmpty | Создает стандартный элемент для пустой последовательности. |
| ElementAt | Возвращает элемент последовательности по указанному индексу. |
| ElementAtOrDefault | Возвращает элемент по указанному индексу или стандартный элемент (если индекс вышел за пределы диапазона). |
| First | Возвращает первый элемент последовательности. |
| FirstOrDefault | Возвращает первый элемент последовательности или стандартный элемент (если нужный элемент не найден). |
| Last | Возвращает последний элемент последовательности. |
| LastOrDefault | Возвращает последний элемент последовательности или стандартный элемент (если нужный элемент не найден). |
| Single | Возвращает единственный элемент последовательности. |
| SingleOrDefault | Возвращает единственный элемент последовательности или стандартный элемент (если нужный элемент не найден). |
| **Равенство** |  |
| SequenceEqual | Проверяет эквивалентность двух последовательностей. |
| **Создание** |  |
| Empty | Создает пустую последовательность. |
| Range | Создает последовательность в соответствии с заданным диапазоном. |
| Repeat | Создает последовательность, повторяя значение заданное количество раз. |
| **Группировка** |  |
| GroupBy | Группирует элементы последовательности указанным образом. |
| **Присоединение** |  |
| GroupJoin | Выполняет группированное соединение двух последовательностей. |
| Join | Выполняет внутреннее соединение двух последовательностей. |
| **Упорядочение** |  |
| OrderBy | Упорядочивает элементы последовательности по заданным значениям в порядке возрастания. |
| OrderByDescending | Упорядочивает элементы последовательности по заданным значениям в порядке убывания. |
| ThenBy | Упорядочивает элементы уже упорядоченной последовательности в порядке возрастания. |
| ThenByDescending | Упорядочивает элементы уже упорядоченной последовательности в порядке убывания. |
| Reverse | Зеркально отображает порядок расположения элементов в последовательности. |
| **Разделение на части** |  |
| Skip | Возвращает последовательность, в которой указанное число элементов пропущено. |
| SkipWhile | Возвращает последовательность, в которой пропущены элементы, не соответствующие указанному условию. |
| Take | Возвращает последовательность, в которую включается указанное число элементов. |
| TakeWhile | Возвращает последовательность, в которую включаются элементы, соответствующие указанному условию. |
| **Проекция** |  |
| Select | Создает проекцию части последовательности. |
| SelectMany | Создает проекцию части последовательности по принципу «один ко многим». |
| **Кванторы** |  |
| All | Определяет соответствие всех элементов последовательности указанным условиям. |
| Any | Определяет, есть ли в последовательность элементы, удовлетворяющие указанным условиям. |
| Contains | Определяет, есть ли в последовательности указанный элемент. |
| **Ограничение** |  |
| Where | Сортирует члены последовательности. |
| **Настройка** |  |
| Distinct | Возвращает последовательность без повторяющихся элементов. |
| Except | Возвращает последовательность, представляющую собой разность двух других последовательностей. |
| Intersect | Возвращает последовательность, представляющую собой пересечение двух других последовательностей. |
| Union | Возвращает последовательность, представляющую собой объединение двух других последовательностей. |

1. Для использования System.Linq необходимо подключить сборку System.Core.dll. [↑](#footnote-ref-1)