Linq запросы

otus.ru

Проверить, идет ли запись

Напишите «+» в чат, если меня слышно и видно

Тема урока

Linq-запросы

Нилов Павел

Fullstack разработчик компании Волховец

Преподаватель курса C# Basic, C# Professional

Контакты:t.me/@NilovPavel

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат

Вопросы вижу в чате,
могу ответить не сразу

Маршрут вебинара

1. Что такое linq
2. Провайдеры данных
3. Варианты синтаксиса
4. Декларативный синтаксис
5. Метод-синтаксис
6. Практика

Цели вебинара

1. Узнать, что такое linq
2. Познакомиться с провайдерами данных linq-запросов
3. Познакомиться с синтаксисом запросов linq в декларативном синтаксисе и в синтаксисе Fluent API

Что такое linq?

Что такое linq?

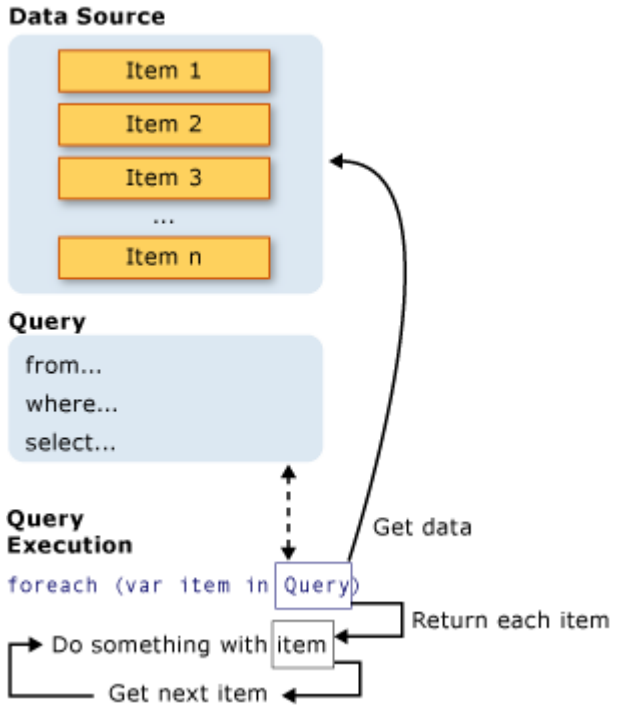
Linq(**Language Integrated Query**) – это синтаксис в .net-языках, спроектированный для работы с источниками данных, таких как коллекции, базы данных, XML-документы и пр.

Linq-запрос – это способ выполнения операций с данными, который интегрирован в язык.

Из чего состоит linq-запрос

Все linq–запросы подразумевают в себе 3 следующие действия:

1. Выбор источника данных
2. Описание тела запроса
3. Выполнение запроса



Отложенное и немедленное выполнение

Отложенное выполнение (Deferred Execution) в LINQ — это механизм, при котором запрос не выполняется в момент его создания, а откладывается до тех пор, пока к данным не будет фактического обращения.

Существуют методы, которые вызывают **немедленное выполнение (Immediate Execution)** запроса и загрузку данных в память. В основном это агрегатные функции и методы возвращения коллекции.

Классификационная таблица

Провайдеры данных linq-запросов

Провайдеры данных, доступные в linq

Все запросы можно отправлять в любой источник данных, реализующий интерфейс IEnumerable<T>, такие как:

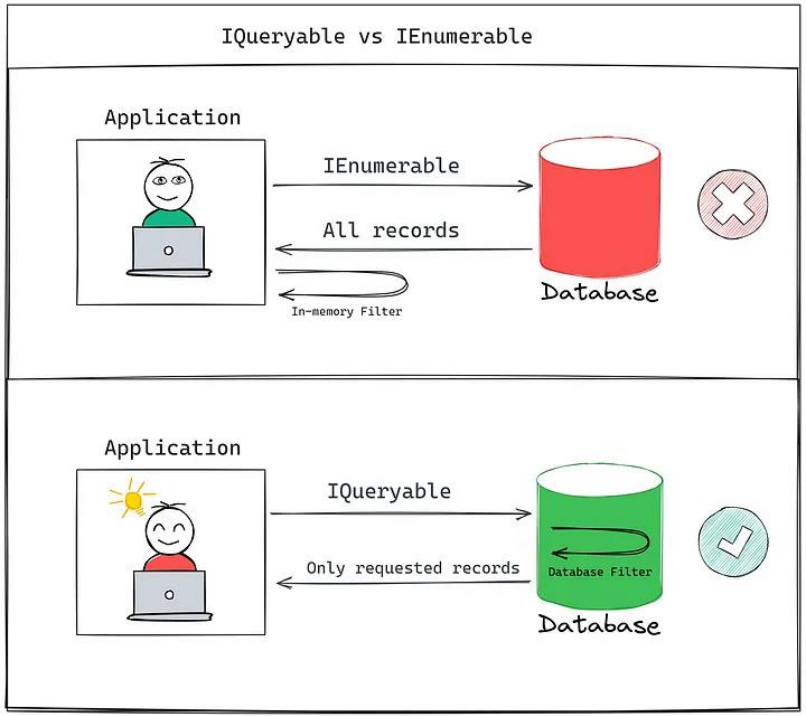
1. **LINQ to Objects**: применяется для работы с массивами и коллекциями
2. **LINQ to Entities**: используется при обращении к базам данных через технологию Entity Framework
3. **LINQ to XML**: применяется при работе с файлами XML
4. **LINQ to DataSet**: применяется при работе с объектом DataSet
5. **Parallel LINQ (PLINQ)**: используется для выполнения параллельных запросов

Интерфейс IQueryable<T>

IQueryable<T> — это интерфейс, который используется в LINQ для создания запросов к удаленным источникам данных.

Используйте *IQueryable*, когда работаете с удаленными источниками данных, например, с базами данных или веб-сервисами.

Используйте *IEnumerable*, когда данные уже находятся в памяти, например, в списке, массиве или другой коллекции.



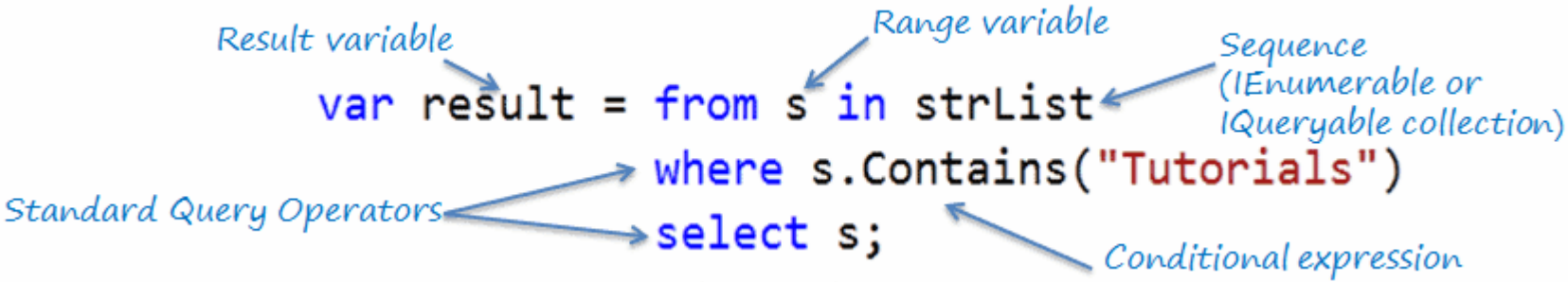
Варианты синтаксиса

Варианты написания запросов

Существует 2 формы написания linq-запросов:

1. Декларативный синтаксис/синтаксис запросов/SQL-подобный синтаксис
2. Fluent API/Метод-синтаксис/Недекларативный синтаксис

Декларативный синтаксис



При использовании декларативного синтаксиса всегда используются 3 предложения:

1. from – указывается псевдоним элемента источника данных;
2. in - указывается источник данных;
3. select/group – возвращаемые элементы данных.

Fluent-синтаксис

При использовании метод-синтаксиса используются методы расширения интерфейса IEnumerable<T>.

Просмотреть исходный код методов можно по ссылке.

```
var query =
        from n in numbers
        where n > 0
        select n;
```

```
var query =
numbers
.Where(item => item > 0)
.Select(item=>item);
```

Особенности декларативного синтаксиса

Декларативный подход не поддерживает:

- Агрегатные функции: `Sum()`, `Count()`, `Max()`, `Min()`, `Average()`;
- Методы пагинации: `Skip()`, `SkipWhile()`, `Take()`, `TakeWhile()`;
- Методы для работы с порядком элементов: `ThenBy()`, `ThenByDescending()`;
- Методы для работы с множествами: `Distinct()`, `Union()`, `Intersect()`, `Except()`;
- Методы группового соединения: `GroupJoin()`;
- Экзистенциальные запросы: `Any()`, `All()`, `Contains()`.

Особенности Fluent API

Метод-синтаксис не поддерживает:

- Запросы с ключевым словом `let`;
- Запросы с ключевым словом `into`.

Декларативный синтаксис

Ключевые слова

При использовании декларативного синтаксиса используются следующие ключевые слова:

- from – где указывается элемент последовательности(обязателен); Используется с in;
- in - источник данных(обязателен);
- where – задает фильтрацию элементов последовательности(опционально);
- select – возвращает проекцию данных(обязательно, либо group);
- group – возвращает объекты типа IGrouping<TKey,TElement>; используется с by; (обязательно, либо select);
- by – указывает параметр, по которому необходимо сгруппировать члены последовательности;

Ключевые слова

При использовании декларативного синтаксиса используются следующие ключевые слова:

- let – определяет переменную и присваивает ей значение, рассчитанное на основе значений данных;
- into – служит для создания временного идентификатора для хранения результата из group;
- join – объединяет различные последовательности, имеющих отношения в объектной модели; Используется с on;
- on – задает ключи, по которым необходимо сопоставить коллекции в join;
- equals – сравнивает значения в выражениях запроса; используется с join on;
- orderby – сортирует последовательность значений, используя компаратор по умолчанию;
- ascending/descending – сортирует последовательность значений, используя компаратор по умолчанию.

Правила описания запроса

1. Выражение должно начинаться с ключевого слова from и должно заканчиваться select или group.
2. Между первым и последним предложением могут находится where, orderby, join, let. Может быть дополнительный from, может быть into, join, group.
3. Выражение может возвращать 2 типа: var или IEnumerable<T>.
4. Запросы могут иметь вложенность.

Пример запроса

Пример запроса выглядит следующим образом:

```
var query =
from item in source
where item.Department.DepartmentName.Equals("ИТ")
orderby item.Name /*ascending*/ /*descending*/
select item;
```

Метод-синтаксис

Метод синтаксис

Все методы расширения предоставляются интерфейсом IEnumerable<T>.

Список находится в исходном коде класса Enumerable.cs, расширяющим интерфейс.

Все методы fluent API на основе LINQ:

1. Принимают обобщенный делегат (анонимные функции) в качестве параметра.
2. Возвращают другую последовательность или одно значение.
3. Могут принимать еще одну коллекцию как параметр(Join).

Методы расширения IEnumerable<T>

Функционал метод-синтаксиса идентичен функционалу декларативного синтаксиса с той разницей, что:

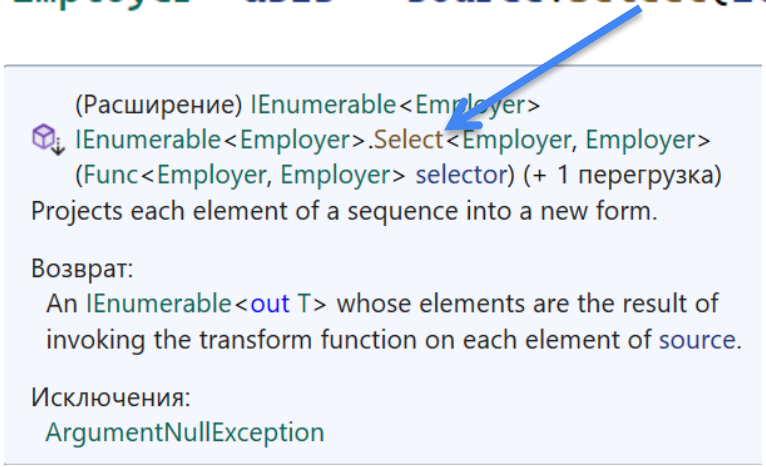
1. Нет ключевых слов: from, in, let, into.
2. Ключевые слова операторов запросов созвучны по написанию, но работают посредством обобщенных делегатов: Action, Predicate, Func.

Пример:

```
var query = source.Where(item => item.Id > 1).Select(item
=> item);
```

Пример синтаксиса

```
IEnumerable<Employer> asIs = source.Select(item => item);
```




```
Func<Employer, Employer> func =
            (item) => item;

asIs = source.Select(func);
```

Практика

Решение задач

1. Напишите запрос к коллекции int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }, который вернет все числа кратные 3.
2. Напишите запрос к коллекции из п.1, который вернет объект(любого типа), содержащий число из коллекции, например Student, в котором StudentId = n(n – число из коллекции п.1);
3. Напишите запрос к коллекции

```
List<Student> students = new List<Student>
{
    new Student { StudentId = 1, Name = "Ivanov", GroupId = 1 },
    new Student { StudentId = 2, Name = "Petrov", GroupId = 2 },
    new Student { StudentId = 3, Name = "Sidorov", GroupId = 1 },
};
```

запрос, который сгруппирует студентов по GroupId.

4. Напишите запрос, выводящий количество студентов из п.3
5. Напишите запрос, сортирующий элементы коллекции из п.3 по убыванию идентификатора студента.

Рефлексия

Отправьте цифры, какие из целей достигнуты

1. Узнали, что такое linq
2. Познакомились с провайдерами данных linq-запросов
3. Познакомились с синтаксисом запросов linq в декларативном синтаксисе и в синтаксисе Fluent API

Вопросы?

Задаем
вопросы в чат

Ставим “–”,
если вопросов нет