.NET 6: LINQ improvements

New features

Gashkov Roman **@macdeath667**

A little before the start

Data used in the code examples

```
public static IEnumerable<Cyclist> Cyclists => new List()
{
    new Cyclist {Id = 1, Name = "Gregor", Age = 33},
    new Cyclist {Id = 2, Name = "Roman", Age = 32},
    new Cyclist {Id = 3, Name = "Roma", Age = 28},
    new Cyclist {Id = 4, Name = "Pavel", Age = 29},
    new Cyclist {Id = 5, Name = "Ghost", Age = 33}
};
```

MinBy() method

```
Cyclist oldWay = source.OrderBy(person => person.Age).First();

Cyclist newWay = source.MinBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

MinBy() method

```
Cyclist oldWay = source.OrderBy(person => person.Age).First();

Cyclist newWay = source.MinBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

MinBy() method

```
Cyclist oldWay = source.OrderBy(person => person.Age).First();
```

```
Cyclist newWay = source.MinBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

MaxBy() method

```
Cyclist oldWay = source.OrderByDescending(person => person.Age).First();

Cyclist newWay = source.MaxBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

MaxBy() method

```
Cyclist oldWay = source.OrderByDescending(person => person.Age).First();

Cyclist newWay = source.MaxBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

MaxBy() method

```
Cyclist oldWay = source.OrderByDescending(person => person.Age).First();
```

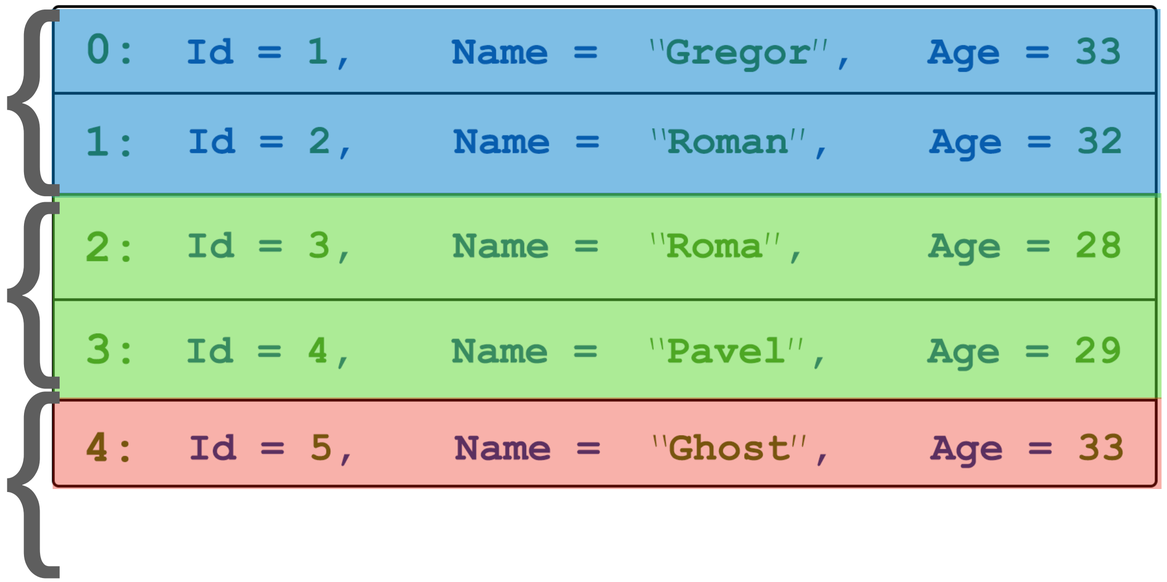
```
Cyclist newWay = source.MaxBy(person => person.Age);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Chunk() method

```
//Before we can do like this:

IEnumerable<IEnumerable<Cyclist>> cluster = source
     .Select((x, i) => new {Index = i, Value = x})
     .GroupBy(x => x.Index / chunkSize)
     .Select(x => x.Select(v => v.Value));
```



Chunk() method

```
IEnumerable<IEnumerable<Cyclist>> cluster = source.Chunk(chunkSize);
```

Chunk() method

```
IEnumerable<IEnumerable<Cyclist>> cluster = source.Chunk(chunkSize);

//Congratulations! You are Great!
```

*By() methods: Operating data

```
IEnumerable<Cyclist> evenAgedPeople = source.Where(person => person.Age % 2 == 0);
```

0:	Id = 2,	Name = "Roman",	Age = 32
1:	Id = 3,	Name = "Roma",	Age = 28

```
IEnumerable<Cyclist> personAbove30 = source.Where(person => person.Age > 30);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 5,	Name = "Ghost",	Age = 33

*By() methods: UnionBy()

```
//What we did before:

IEnumerable<Cyclist> union = evenAgedPeople.Union(personAbove30);
```

*By() methods: UnionBy()

```
//or another sample what we did before:

IEnumerable<Cyclist> union = evenAgedPeople.Union(personAbove30, new PersonByAgeComparer());
```

*By() methods: UnionBy()

```
//or another sample what we did before:

IEnumerable<Cyclist> union = evenAgedPeople.Union(personAbove30, new PersonByAgeComparer());
```

```
class PersonByAgeComparer : IEqualityComparer<Cyclist>
{
    public bool Equals(Cyclist x, Cyclist y)
    {
        if (ReferenceEquals(x, null)) return false;
        if (ReferenceEquals(y, null)) return false;
        return x.Age == y.Age;
    }

    public int GetHashCode(Cyclist obj) => HashCode.Combine(obj.Age);
}
```

*By() methods: UnionBy()

```
//What we did before:

IEnumerable<Cyclist> union = evenAgedPeople.Union(personAbove30, new PersonByAgeComparer());
```

0:	Id = 2,	Name = "Roman",	Age = 32
1:	Id = 3,	Name = "Roma",	Age = 28

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 5,	Name = "Ghost",	Age = 33

Result:

0:	Id = 2,	Name = "Roman",	Age = 32
1:	Id = 3,	Name = "Roma",	Age = 28
2:	Id = 1,	Name = "Gregor",	Age = 33

*By() methods: UnionBy()

```
IEnumerable<Cyclist> unionBy = evenAgedPeople.UnionBy(personAbove30, x => x.Age);
```

*By() methods: UnionBy()

```
IEnumerable<Cyclist> unionBy = evenAgedPeople.UnionBy(personAbove30, x => x.Age);

//Congratulations! You are Great Again!
```

*By() methods: UnionBy()

```
var unionBy = evenAgedPeople.UnionBy(personAbove30, x => x.Age, new AgeComparer());
```

*By() methods: IntersectBy()

```
IEnumerable<Cyclist> intersect = evenAgedPeople.Intersect(personAbove30, new PersonByAgeComparer());

IEnumerable<Cyclist> intersectBy = evenAgedPeople.IntersectBy(personAbove30.Select(x=>x.Age),x=>x.Age);
```

0:	Id = 2,	Name = "Roman",	Age = 32
1:	Id = 3,	Name = "Roma",	Age = 28

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 5,	Name = "Ghost",	Age = 33

Result:

0:	Id = 2,	Name = "Roman",	Age = 32

*By() methods: ExceptBy()

```
IEnumerable<Cyclist> except = evenAgedPeople.Except(personAbove30, new PersonByAgeComparer());

IEnumerable<Cyclist> exceptBy = evenAgedPeople.ExceptBy(personAbove30.Select(x=>x.Age),x=>x.Age);
```

```
0:  Id = 2,  Name = "Roman",  Age = 32
1:  Id = 3,  Name = "Roma",   Age = 28
```

```
0:  Id = 1,  Name = "Gregor",  Age = 33
1:  Id = 2,  Name = "Roman",   Age = 32
2:  Id = 5,  Name = "Ghost",   Age = 33
```

Result:

```
0:  Id = 3,  Name = "Roma",  Age = 28
```

*By() methods: DistinctBy()

```
IEnumerable<Cyclist> distinct = personAbove30.Distinct(new PersonByAgeComparer());

IEnumerable<Cyclist> distinctBy = personAbove30.DistinctBy(x => x.Age);
```

```
0:  Id = 1,    Name =  "Gregor",   Age = 33
1:  Id = 2,    Name =  "Roman",    Age = 32
2:  Id = 5,    Name =  "Ghost",    Age = 33
```

Index and Ranges

```
var array = new int[] { 1, 2, 3, 4, 5 };

var thirdItem = array[2];  // array[2]
var lastItem = array[^1];  // array[new Index(1, fromEnd: true)]

var slice1 = array[2..^3]; // array[new Range(2, new Index(3, fromEnd: true))]
var slice2 = array[..^3];  // array[Range.EndAt(new Index(3, fromEnd: true))]
var slice3 = array[2..];   // array[Range.StartAt(2)]
var slice4 = array[..];    // array[Range.All]
```

Index and Ranges

```
Cyclist secondLastPersonOld = source.TakeLast(2).FirstOrDefault();
Cyclist secondLastPerson = source.ElementAt(^2);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> take3PeopleOld = source.Take(3);
IEnumerable<Cyclist> take3People = source.Take(..3);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> skip1PersonOld = source.Skip(1);
IEnumerable<Cyclist> skip1Person = source.Take(1..);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> take3Skip1PeopleOld = source.Take(3).Skip(1);
IEnumerable<Cyclist> take3Skip1People = source.Take(1..3);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> takeLast2PeopleOld = source.TakeLast(2);
IEnumerable<Cyclist> takeLast2People = source.Take(^2..);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> skipLast3PeopleOld = source.SkipLast(3);
IEnumerable<Cyclist> skipLast3People = source.Take(..^3);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Index and Ranges

```
IEnumerable<Cyclist> takeLast3SkipLast2Old = source.TakeLast(3).SkipLast(2);
IEnumerable<Cyclist> takeLast3SkipLast2 = source.Take(^3..^2);
```

0:	Id = 1,	Name = "Gregor",	Age = 33
1:	Id = 2,	Name = "Roman",	Age = 32
2:	Id = 3,	Name = "Roma",	Age = 28
3:	Id = 4,	Name = "Pavel",	Age = 29
4:	Id = 5,	Name = "Ghost",	Age = 33

Single/Last/FirstOrDefault(T default) overloads

```
IEnumerable<Cyclist> emptyCyclists = new List<Cyclist>();

Cyclist firstOrDefault = emptyCyclists.FirstOrDefault();
Cyclist overloadedFirstOrDefault = emptyCyclists.FirstOrDefault(Cyclist.Empty);

//Old FirstOrDefault: null;
//Overloaded FirstOrDefault: Empty;
```

```
public class Cyclist
{
    public static readonly Cyclist Empty = new {Id = -1, Name = "None", Age = -1};
    …
}
```

Single/Last/FirstOrDefault(T default) overloads

```
int old = enumerable.Where(x => x.Age < 10).Select(x => x.Age).FirstOrDefault();
int overloaded = enumerable.Where(x => x.Age < 10).Select(x => x.Age).FirstOrDefault(-1);


//Old FirstOrDefault: 0;
//Overloaded FirstOrDefault: -1;
```

```
0:  Id = 1,    Name =  "Gregor",   Age = 33
1:  Id = 2,    Name =  "Roman",    Age = 32
2:  Id = 3,    Name =  "Roma",     Age = 28
3:  Id = 4,    Name =  "Pavel",    Age = 29
4:  Id = 5,    Name =  "Ghost",    Age = 33
```

Zip(…) takes 3 arguments

```
IEnumerable<string> allNames = source.Select(x=>x.Name);
IEnumerable<int> allAges = source.Select(x => x);

//zip two collections
IEnumerable<(string name, int Age)> zip2 = allNames.Zip(allAges);
```

Zip(...) takes 3 arguments

```
IEnumerable<string> allNames = source.Select(x=>x.Name);
IEnumerable<int> allAges = source.Select(x => x.Age);
IEnumerable<int> ids = source.Select(x => x.Id);
```

```
//old way to zip three collections
IEnumerable<(int Id, string name, int Age)> zip3Old = ids
            .Zip(allNames)
            .Zip(allAges)
            .Select(x=>(x.First.First, x.First.Second, x.Second));
```

Zip(…) takes 3 arguments

```
IEnumerable<string> allNames = source.Select(x=>x.Name);
IEnumerable<int> allAges = source.Select(person => person.Age);
IEnumerable<int> ids = source.Select(person => x.Id);

//new way to zip three collections
IEnumerable<(int Id, string name, int Age)> zip3 = ids.Zip(allNames, allAges);
```

TryGetNotEnumeratedCount()

```
var oldWayCount = source.Count();

bool doneWithoutEnumerating = source.TryGetNonEnumeratedCount(out var sourceCount);
```

Bonus track

```
public static IEnumerable<Cyclist> InfinitiveCyclistsEnumerable
{
    get
    {
        while (true)
        {
            yield return new Cyclist {Id = 1, Name = "Gregor", Age = 33};
            yield return new Cyclist {Id = 2, Name = "Roman", Age = 32};
            yield return new Cyclist {Id = 3, Name = "Roma", Age = 28};
            yield return new Cyclist {Id = 4, Name = "Pavel", Age = 29};
            yield return new Cyclist {Id = 5, Name = "Ghost", Age = 33};
        }
    }
}
```

Bonus track

```
public static IEnumerable<Cyclist> InfinitiveCyclistsEnumerable
{
    get
    {
        while (true)
        {
            yield return new Cyclist {Id = 1, Name = "Gregor", Age = 33};
            yield return new Cyclist {Id = 2, Name = "Roman", Age = 32};
            yield return new Cyclist {Id = 3, Name = "Roma", Age = 28};
            yield return new Cyclist {Id = 4, Name = "Pavel", Age = 29};
            yield return new Cyclist {Id = 5, Name = "Ghost", Age = 33};
        }
    }
}

//You will iterate over such a collection until the death tear us apart
```

References

Docs

- https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6
- https://devblogs.microsoft.com/dotnet/announcing-net-6/

Examples

- https://github.com/MacDeath667/.NET-6-LINQ-review

Thank you!