Enumerable.Any Método

Referência

Definição

Namespace: System.Linq

Assembly: System.Linq.dll

Determina se qualquer elemento de uma sequência existe ou atende a uma condição.

Sobrecargas

Any<TSource>(IEnumerable<TSource>)	Determina se uma sequência contém elementos.
Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)	Determina se algum elemento de uma sequência atende a uma condição.

Any<TSource>(IEnumerable<TSource>)

Determina se uma sequência contém elementos.

C#

```
public static bool Any<TSource> (this
System.Collections.Generic.IEnumerable<TSource> source);
```

Parâmetros de tipo

`TSource`

O tipo dos elementos de `source`.

Parâmetros

`source` IEnumerable<TSource>

O IEnumerable<T> a ser verificado se está vazio.

Retornos

Boolean

`true` se a sequência de origem contiver elementos; caso contrário, `false`.

Exceções

ArgumentNullException

`source` é `null`.

Exemplos

O exemplo de código a seguir demonstra como usar Any para determinar se uma sequência contém elementos.

C#

```
List<int> numbers = new List<int> { 1, 2 };
bool hasElements = numbers.Any();

Console.WriteLine("The list {0} empty.",
    hasElements ? "is not" : "is");

// This code produces the following output:
//
// The list is not empty.
```

O valor booliano que o Any<TSource>(IEnumerable<TSource>) método retorna normalmente é usado no predicado de uma `where` cláusula (`Where` cláusula em Visual Basic) ou uma chamada direta ao Where<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>) método. O exemplo a seguir demonstra esse uso do `Any` método.

C#

```
class Pet
{
    public string Name { get; set; }
    public int Age { get; set; }
}
class Person
{
```

```
    public string LastName { get; set; }
    public Pet[] Pets { get; set; }
}

public static void AnyEx2()
{
    List<Person> people = new List<Person>
        { new Person { LastName = "Haas",
                       Pets = new Pet[] { new Pet {
Name="Barley", Age=10 },
                                          new Pet {
Name="Boots", Age=14 },
                                          new Pet {
Name="Whiskers", Age=6 }}},
          new Person { LastName = "Fakhouri",
                       Pets = new Pet[] { new Pet { Name =
"Snowball", Age = 1}}},
          new Person { LastName = "Antebi",
                       Pets = new Pet[] { }},
          new Person { LastName = "Philips",
                       Pets = new Pet[] { new Pet { Name =
"Sweetie", Age = 2},
                                          new Pet { Name =
"Rover", Age = 13}} }
        };

    // Determine which people have a non-empty Pet array.
    IEnumerable<string> names = from person in people
                                where person.Pets.Any()
                                select person.LastName;

    foreach (string name in names)
    {
        Console.WriteLine(name);
    }

    /* This code produces the following output:

       Haas
       Fakhouri
       Philips
    */
}
```

Comentários

> **Observação**
>
> Esse método não retorna nenhum elemento de uma coleção. Em vez disso, determina se a coleção contém elementos.

A enumeração de `source` é interrompida assim que o resultado pode ser determinado.

Em Visual Basic sintaxe de expressão de consulta, uma `Aggregate Into Any()` cláusula se traduz em uma invocação de Any.

Confira também

- Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)
- Cláusula Aggregate (Visual Basic)

Aplica-se a

▼ .NET 7 e outras versões

Produto	Versões
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7
.NET Framework	3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0

Any<TSource>(IEnumerable<TSource>, Func<TSource,Boolean>)

Determina se algum elemento de uma sequência atende a uma condição.

C#

```
public static bool Any<TSource> (this
System.Collections.Generic.IEnumerable<TSource> source,
Func<TSource,bool> predicate);
```

Parâmetros de tipo

`TSource`

O tipo dos elementos de `source`.

Parâmetros

`source` IEnumerable<TSource>

Um IEnumerable<T> a cujos elementos o predicado será aplicado.

`predicate` Func<TSource,Boolean>

Uma função para testar cada elemento em relação a uma condição.

Retornos

Boolean

`true` se a sequência de origem não estiver vazia e pelo menos um dos elementos passar no teste no predicado especificado; caso contrário, `false`.

Exceções

ArgumentNullException

`source` ou `predicate` é `null`.

Exemplos

O exemplo de código a seguir demonstra como usar Any para determinar se qualquer elemento em uma sequência satisfaz uma condição.

C#

```
class Pet
{
    public string Name { get; set; }
    public int Age { get; set; }
```

```
    public bool Vaccinated { get; set; }
}

public static void AnyEx3()
{
    // Create an array of Pets.
    Pet[] pets =
        { new Pet { Name="Barley", Age=8, Vaccinated=true },
          new Pet { Name="Boots", Age=4, Vaccinated=false },
          new Pet { Name="Whiskers", Age=1, Vaccinated=false }
};

    // Determine whether any pets over age 1 are also unvaccina-
ted.
    bool unvaccinated =
        pets.Any(p => p.Age > 1 && p.Vaccinated == false);

    Console.WriteLine(
        "There {0} unvaccinated animals over age one.",
        unvaccinated ? "are" : "are not any");
}

// This code produces the following output:
//
//  There are unvaccinated animals over age one.
```

Comentários

> **Observação**
>
> Esse método não retorna nenhum elemento de uma coleção. Em vez disso, ele determina se qualquer elemento de uma coleção satisfaz uma condição.

A enumeração de `source` é interrompida assim que o resultado pode ser determinado.

Em Visual Basic sintaxe de expressão de consulta, uma `Aggregate Into Any()` cláusula é convertida em uma invocação de Any.

Confira também

- Cláusula Aggregate (Visual Basic)

Aplica-se a

▼ .NET 7 e outras versões

Produto	Versões
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7
.NET Framework	3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0