Converting from IEnumerable to List [duplicate]

231

This question already has an answer here:
Casting IEnumerable<T> to List<T> *5 answers*

I want to convert from `IEnumerable<Contact>` to `List<Contact>`. How can I do this?

23

c# list ienumerable

edited Aug 18 '15 at 21:49
Koterpillar
5,806 2 20 32

asked Oct 1 '11 at 2:25
kartal
6,118 29 84 136

marked as duplicate by Jannie Theunissen, JOBG, freefaller, nafas, Cheesebaron Feb 17 '17 at 17:41

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

5 Answers

385

You can do this very simply using LINQ.

Make sure this using is at the top of your C# file:

```
using System.Linq;
```

```
IEnumerable<int> enumerable = Enumerable.Range(1, 300);
List<int> asList = enumerable.ToList();
```

edited Sep 7 '18 at 20:14
jpaugh
4,081 3 26 70

answered Oct 1 '11 at 2:28
vcsjones
108k 23 252 258

42 It is important to note that your solution works for the generic version of `IEnumerable` . The answer from user pickles below handles the non-generic version. – mkmurray Mar 20 '13 at 19:49

11 Thanks for "using System.Linq;" – mili Apr 6 '15 at 10:09

3 Thank you @mkmurray I need to pay close attention to find the "pickles" :D – fabriciorissetto Oct 29 '15 at 0:43

2 I wonder if Microsoft changed this. I just tried this very example, and there is no ToList() method in IEnumerable. Using VS 2015 and .NET 4.6.1 . – James Dec 22 '16 at 20:20

2 @James it's an extension method. You may need the "using", but no, they did not change this. – vcsjones Dec 22 '16 at 20:27

157

In case you're working with a regular old `System.Collections.IEnumerable` instead of `IEnumerable<T>` you can use `enumerable.Cast<object>().ToList()`

answered Oct 1 '11 at 2:30
cordialgerm
6,883 4 22 47

Love it. Thank you!<3 – karlingen Dec 30 '18 at 20:02

If you're using an implementation of `System.Collections.IEnumerable` you can do like following to convert it to a `List` . The following uses [Enumerable.Cast](#) method to convert `IEnumberable` to a Generic `List` .

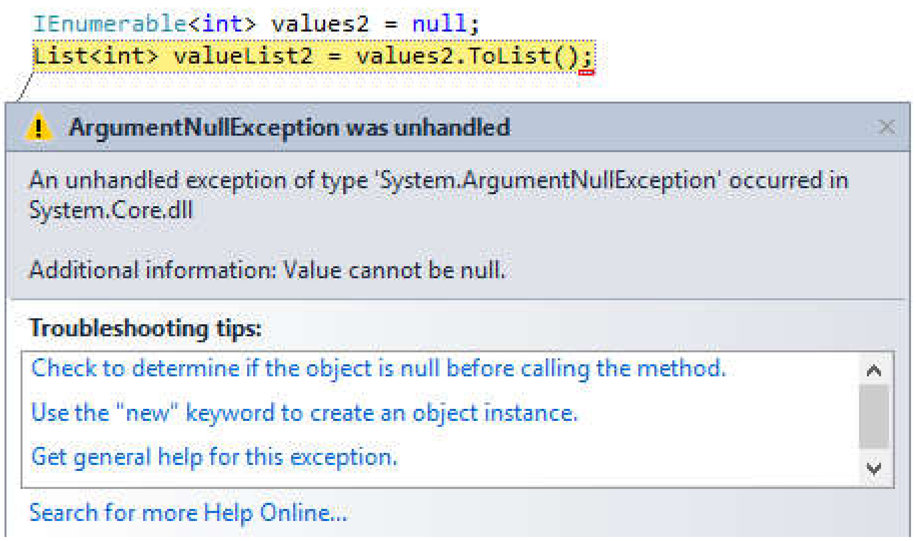
```
List<string> provinces = _provinces.Cast<string>().ToList();
```

If you're using Generic version `IEnumerable<T>` , The conversion is straight forward. Since both are generics, you can do like below,

```
IEnumerable<int> values = Enumerable.Range(1, 10);
List<int> valueList = values.ToList();
```

But if the `IEnumerable` is null, when you try to convert it to a `List` , you'll get `ArgumentNullException` saying Value cannot be null.

```
IEnumerable<int> values2 = null;
List<int> valueList2 = values2.ToList();
```



```
IEnumerable<int> values2 = null;
List<int> valueList2 = values2.ToList();
```

ArgumentNullException was unhandled

An unhandled exception of type 'System.ArgumentNullException' occurred in System.Core.dll

Additional information: Value cannot be null.

Troubleshooting tips:

Check to determine if the object is null before calling the method.

Use the "new" keyword to create an object instance.

Get general help for this exception.

Search for more Help Online...

Therefore as mentioned in the other answer, remember to do a `null` check before converting it to a `List` .

edited May 23 '17 at 12:03
Community ♦

answered Jul 18 '15 at 4:10
Nipuna

5

```
List<int> list=new List<int>();

IEnumerable<int> enumerable =Enumerable.Range(1, 300);

foreach (var item in enumerable )
{
  list.add(item);
}
```

answered Nov 26 '14 at 17:45
NourAldienArabian
79 1 4

It's quite better to use "AddRange(enumerable)" function or List(enumerable) constructor. – QtRoS Sep 22 '15 at 14:41

5

I use an extension method for this. My extension method first checks to see if the enumeration is null and if so creates an empty list. This allows you to do a foreach on it without explicitly having to check for null.

Here is a very contrived example:

```
IEnumerable<string> stringEnumerable = null;
StringBuilder csv = new StringBuilder();
stringEnumerable.ToNonNullList().ForEach(str=> csv.Append(str).Append(","));
```

Here is the extension method:

```
public static List<T> ToNonNullList<T>(this IEnumerable<T> obj)
{
    return obj == null ? new List<T>() : obj.ToList();
}
```

edited Jan 4 '17 at 5:33 answered Aug 6 '13 at 21:17