The results are in! See what nearly 90,000 developers picked as their most loved, dreaded, and desired coding languages and more in the 2019 Developer Survey.

LINQ to SQL and Null strings, how do I use Contains?

Ask Question



Here is the query

22

from a in this._addresses
where a.Street.Contains(street) || a.StreetAdditional.Contains(streetAdditional)
select a).ToList<Address>()



if both properties in the where clause have values this works fine, but if for example, a.StreetAdditional is null (Most of the times), I will get a null reference exception.

Is there a work around this?

Thanks,

c# .net linq-to-sql

asked Jun 10 '09 at 17:49



Oakcool 1,017 1 10 29

Did you get an exception? Or are you speculating than an exception is possible? – Amy B Jun 10 '09 at 18:07

I got the exception. - Oakcool Jun 12 '09 at 18:58

If you've got a NullReferenceException for that, you aren't doing a LINQ to SQL query. — Pavel Minaev Nov 29 '09 at 5:17

You could also think about not allowing Street or StreetAdditional to be null. If your db supports default values you could default these to an empty string, set the flag to disallow nulls and obviate the need to null check. — Tod Dec 13 '11 at 18:39

On a side note, the query does work in LinqPad. How is this possible? Why is there such a difference in Linq to SQL behavior between the two tools? – dpant Nov 13 '15 at 20:14

9 Answers



The most obvious one:

40

from a in this._addresses
where (a.Street != null && a.Street.Contains(street)) || (a.StreetAc
a.StreetAdditional.Contains(streetAdditional))
select a).ToList<Address>()



Alternatively you could write an extension method for Contains that accepts a null argument without error. Some might say that it is not so pretty to have such a method, because it looks like a normal method call, but is allowed for null values (thereby setting aside normal null-checking practices).

answered Jun 10 '09 at 17:52



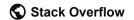
driis

124k 41 239 32

- 1 It looks like its working Thanks Oakcool Jun 10 '09 at 19:20
- 4 Custom extension methods aren't usable in LINQ to SQL. Pavel Minaev Nov 29 '09 at 5:16

Home

PUBLIC



Tags

Users

Jobs

Teams

Q&A for work





One thing to note is that the null should be evaluated first.



```
where (**a.Street != null** && a.Street.Contains(street)) || (a.Stre
&& a.StreetAdditional.Contains(streetAdditional))
select a).ToList<Address>
```

()





sujith karivelil 23.7k 6 30 61

answered Feb 22 '16 at 18:54



user5636696

41 1



I'd use the null-coalescing operator...

46

```
(from a in this._addresses
where (a.Street ?? "").Contains(street) || (a.StreetAdditional ??
"").Contains(streetAdditional)
select a).ToList<Address>()
```

edited Dec 14 '13 at 9:21

answered Jun 10 '09 at 18:02



Yuliy

13.7k 4 34 44

This is a cleaner approach. Perfect for when dealing with an IQueryable<T> – DanKodi Jul 28 '14 at 4:50



You might want to check to make sure the variables street and streetAdditional are not null. I just ran across the same problem and setting them to an empty string seemed to solve my problem.



```
street = street ?? "";
streetAdditional = streetAdditional ?? "";
from a in this._addresses
where a.Street.Contains(street) || a.StreetAdditional.Contains(street)
select a).ToList<Address>()
```

answered Nov 29 '09 at 4:37





I don't think SqlServer gave you a null exception. If it did, then this code is clearly not running though LinqToSql (as you've tagged the question).



string. Contains would be translated to sql's like, which has no trouble at all with null values.

answered Jun 10 '09 at 18:04





I would create an extension method to return an empty sequence if null and then call contains method.

4

```
 \begin{array}{llll} \textbf{public static IEnumerable} < \texttt{T} > & \textbf{EmptyIfNull} < \texttt{T} > (\textbf{this IEnumerable} < \texttt{T} > & \texttt{pSeq} \\ \end{array}
```



```
return pSeq ?? Enumerable.Empty<T>();
from a in this. addresses
where a.Street.Contains(street) ||
      a.StreetAdditional.EmptyIfNull().Contains(streetAdditional)
select a).ToList<Address>()
```

answered Jun 10 '09 at 17:54



Vasu Balakrishnan

Looks somewhat strange because a null -objects seems to be able to invoke member functions. - Dario Jun 10 '09 at 17:58

Upvote for the use of the ?? operator. That's what I would have suggested as well. If you feel that the extension method makes the guery look strange you could do without the extension method. – Jeroen Huinink Jun 10 '09 at 18:04



You must check first if StreetAdditional is null.

Try



where a.Street.Contains(street) || ((a != null) && a.StreetAdditional.Contains(streetAdditional))

This works because && is a shortcut-operator and if a != null yields false, the second expression with the null -value won't be evaluated since the result will be false anyway.

answered Jun 10 '09 at 17:54





from a in this._addresses
where a.Street.Contains(street) || (a.StreetAdditional != null &&
a.StreetAdditional.Contains(streetAdditional)
select a).ToList<Address>()



answered Jun 10 '09 at 17:53



Dimi Takis

4,183 3 23 39



Check to make sure that the properties are not null

1

from a in this._addresses
where (a.Street != null && a.Street.Contains(street)) ||
(a.StreetAdditional != null && a.StreetAdditional.Contains(streetAdd:
select a).ToList<Address>()



If the null check is false, then the second clause after the && will not evaluate.

answered Jun 10 '09 at 17:52



Yaakov Ellis

29.3k 24 109 16