Declaring an implicitly typed variable inside conditional scope and using it outside

Asked 7 years, 8 months ago Active 5 years, 9 months ago Viewed 15k times

5

2

In the simplified code below,

```
if(city == "New York City")
{
  var MyObject = from x in MyEFTable
                     where x.CostOfLiving == "VERY HIGH"
                     select x.*;

}
else
{
  var MyObject = from x in MyEFTable
                     where x.CostOfLiving == "MODERATE"
                     select x.*;

}

  foreach (var item in MyObject)
  {
     Console.WriteLine("<item's details>");
  }
```

The variable MyObject is not accessible outside conditional block. How can I iterate outside the if..else ?

c# foreach var conditional-statements implicit-typing

edited Jan 5 '12 at 23:51

M.Babcock
16.6k 4 44 78

asked Jan 5 '12 at 23:37

FMFF
854 4 22 55

I suppose you could declare the variable outside of the blocks. ChaosPandion Jan 5 '12 at 23:39

1 With `x.*` you mean the construction of an anonymous type, right? If not, why are you insisting on implicit typing? – CodesInChaos Jan 5 '12 at 23:46

Yes I meant anonymous type when I wrote x.*. So in my actual code, the select looks like `select new {columnA, columnB, columnC}` . Sorry for oversimplification. – FMFF Jan 5 '12 at 23:50

3 In *this* case, since the conditional block is only being used to modify the `where` clause of the LINQ query, you can avoid the whole issue: eliminate the `if` block and change the `where` clause to `where x.CostOfLiving == ((city == "New York City") ? "VERY HIGH" : "MODERATE")` . In a more-complex case, replace the `where` clause with an appropriate generic predicate... – Dan J Jan 5 '12 at 23:52

7 Answers

24

Let's clarify your confusing question. The problem is that you have two local variables, each of which has the same "unspeakable" type -- a sequence of anonymous type.

I would change your specific code like this:

```
string cost = city == "NYC" ? "HIGH" : "MODERATE";
var query = from row in table
            where row.Cost == cost
            select new { row.Population, row.Elevation };
```

However, if you still need to maintain the structure of the code as it is for some reason, you can do it like this:

```
static IEnumerable<T> SequenceByExample<T>(T t){ return null; }
...
var query = SequenceByExample(new { Population = 0, Elevation = 0.0 } );
if (whatever)
    query = ...
else
    query = ...
```

This is a variation on a trick called "cast by example" where you give an example of an anonymous type to a generic method. Method type inference then figures out what the return type is, and uses that as the type of the implicitly typed local. At runtime, it does nothing but create a useless object that then gets discarded quickly.

answered Jan 6 '12 at 0:10

563k 153 1097
1968

3 This is, without a doubt, the smartest trick I've seen all year. – Polynomial Oct 10 '12 at 15:30

4

If you're using a named type, just declare a variable with that type before the `if`, but then the question would be trivial.

So I assume you're selecting an anonymous type, so you can't explicitly declare a variable with that type.

Cast by example would work here. But that doesn't feel like a good solution. Probably creating a named type is a better idea.

```
var myObject =Enumerable.Empty<RowType>.Select(row=>select new {columnA, columnB,
columnC});
if(city == "New York City")
{
  myObject= from x in MyEFTable
                    where x.CostOfLiving == "VERY HIGH"
                    select select new {columnA, columnB, columnC};
}
else
{
  myObject = from x in MyEFTable
                    where x.CostOfLiving == "MODERATE"
                    select select new {columnA, columnB, columnC};
}
```

Or in your specific example one could project only after the conditional:

```
IQueryable<RowType> partialQuery;
if(city == "New York City")
    partialQuery=MyEFTable.Where(x=>x.x.CostOfLiving == "VERY HIGH");
else
    partialQuery=MyEFTable.Where(x=>x.x.CostOfLiving == "MODERATE");
var myObject=partialQuery.Select(x=>x.new {columnA, columnB, columnC});
```

Or:

```
  filter=x=>x.x.CostOfLiving == "VERY HIGH";
else
  filter=x=>x.x.CostOfLiving == "MODERATE";
var myObject=MyEFTable.Where(filter).Select(x=>x.new {columnA, columnB, columnC});
```

Or even just:

```
string s;
if(city == "New York City")
  s="VERY HIGH";
else
  s="MODERATE";
var myObject=MyEFTable.Where(x=>x.CostOfLiving == s).Select(x=>x.new {columnA, columnB,
columnC});
```

Which one is appropriate depends on how you simplified your question.

edited Jan 5 '12 at 23:58

answered Jan 5 '12 at 23:41

CodesInChaos

92.1k 15 176 235

3

Try this:

```
var ret = default(object);
```

edited Nov 19 '13 at 12:03

sashkello

10.5k 16 69 99

answered Nov 19 '13 at 11:44

user3006492

36 2

2

try this:

```
System.Linq.IQueryable<MyEFTable Object type> MyObject = null;
if(city == "New York City")
{
```

```
}
else
{
  MyObject = from x in MyEFTable
             where x.CostOfLiving == "MODERATE"
             select x.*;
}

foreach (var item in MyObject)
{
  Console.WriteLine("<item's details>");
}
```

answered Nov 14 '12 at 12:32

Abd Rauf
57 5

1

You will need to declare the variable outside of the scope of the if statement in order to use it in the foreach loop.

If the variable is declared but not initialized outside the if statement it can't be typed implicitly because the compiler won't have an expression to determine the type from.

If it's only going to be used in the foreach loop you can declare it as an IEnumerable.

answered Jan 5 '12 at 23:44

Andrew Kennan
12.5k 3 19 33

1

```
List<MyObject> list = null;

if(city == "New York City")
   list = (from x in MyEFTable  where x.CostOfLiving == "VERY HIGH"
                   select x.*).ToList();
else
   list = (from x in MyEFTable where x.CostOfLiving == "MODERATE"
                   select x.*).ToList();
```

edited Jan 5 '12 at 23:45

answered Jan 5 '12 at 23:40

Tigran
56.2k 6 71 108

0

you'll have to define the MyObject as a var before the condition:

```
var MyObject = from x in MyEFTable
                    where x.CostOfLiving == "SOMETHING THAT'LL RETURN NO ROWS"
                    select x.*;
```

This will assign a schema to the MyObject variable.

Now you can proceed with your condition as:

```
if(city == "New York City")
{
  MyObject = from x in MyEFTable
                    where x.CostOfLiving == "VERY HIGH"
                    select x.*;

}
else
{
  MyObject = from x in MyEFTable
                    where x.CostOfLiving == "MODERATE"
                    select x.*;

}
```

answered Jan 5 '12 at 23:40

Hassan Gulzar
2,105 2 27 58

2 The extra dead query is unnecessary. Just write out the actual type. – ChaosPandion Jan 5 '12 at 23:41

@ChaosPandion I assume the type is anonymous. – CodesInChaos Jan 5 '12 at 23:45