Contents

A common request is to be able to perform an *Include* operation where the *Include* is modified by a where clause. While this is not currently possible, there is another approach that accomplishes almost the same thing. The trick is to to use an anonymous projection.

Anonymous projections

Let say we wanted to write a query that looked something like the following. Note that this code will NOT compile.

```
var query = anEntityManager.Products.Where(p =>p.Category.Name == "Books")
  .(Include("OrderDetails").Where(od => od.Quantity > 5))
```

```
Dim query = anEntityManager.Products.Where(Function(p) p.Category.Name = "Books") _
  .(Include("OrderDetails").Where(Function(od) od.Quantity > 5))
```

What we are try to do is select only those *Products* that are books, and then only *Include* those *OrderDetails*, for these books, where the OrderDetail's Quantity is greater than 5.

The syntax above does NOT work, but we can do something very similar, like this:

```
var query = anEntityManager.Products.Where(p => p.Category.Name == "Books")
      .Select(p => new { Product = p, OrderDetails = p.OrderDetails.Where(od => od.Quantity > 5) });
var results = query.ToList();
var products = results.Select( x => x.Product);
```

```
Dim query = anEntityManager.Products.Where(Function(p) p.Category.Name = _
 "Books").Select(Function(p) New With {Key .Product = p, Key .OrderDetails = _
  p.OrderDetails.Where(Function(od) od.Quantity > 5)})
Dim results = query.ToList()
Dim products = results.Select(Function(x) x.Product)
```

The idea here is in DevForce any query that returns entities causes those entities to be added to the EntityManager's *entity cache*. This is basically, what the *Include* extension method does as well. It bring entities down and "includes" them in the cache, but they are not part of the "result" of the query.

In the example above we actually return an anonymous type that "includes" both *Products* and *OrderDetails*, so both types of entities are added to the cache. But we don't necessarily want an anonymous result, what we possibly want is just the list of Products. Hence, the line after the ToList() call in the example above. In that line we project out just the "Products" that we just queried.