前言

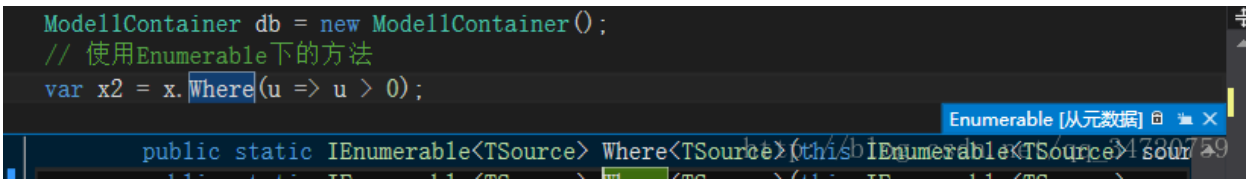
此文章延续自C#中Linq的使用

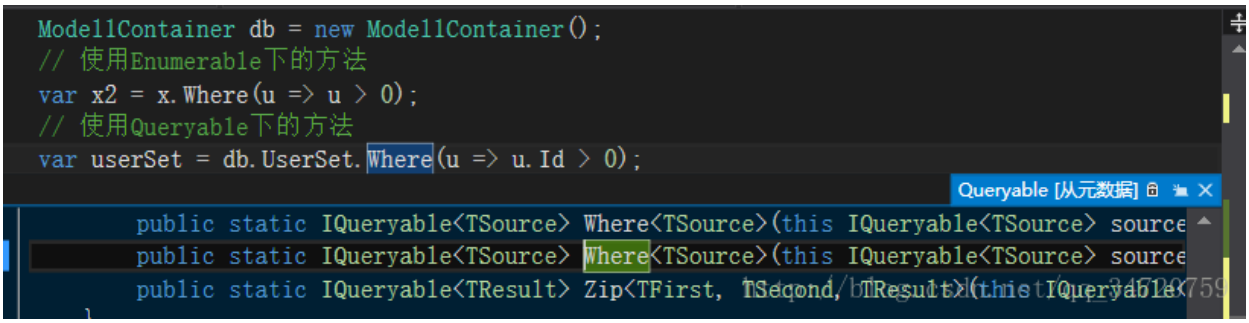
正文

上节讲到Linq可以通过查询方法来使用，Linq中有两个命名空间，分别是System.Linq.Enumerable和System.Linq.Queryable，两个命名空间中几乎拥有相同的方法，使用方式也大致相同，如下：

```
int[] y = { 0, 9, 2, 3, 5 };
Model1Container db = new Model1Container();
// 使用Enumerable下的方法
var x2 = x.Where(u => u > 0);
// 使用Queryable下的方法
var userSet = db.UserSet.Where(u => u.Id > 0);
```

使用方式相同吧，但是两个Where来自不同的命名空间：





接下来分析它们的不同点

使用场合不同

Enumerable适合在内存数据集合中使用（如数组、List等），Queryable适合在离线数据集合中使用（如EF中的dbContext中的DbSet中使用）

这你不需要自己区分，因为当数组等内存集合使用linq查询时自动使用System.Linq.Enumerable下的方法，当EF的dbContext的DbSet使用linq查询时自动使用System.Linq.Queryable下的方法，至于为什么呢？下面的不同点给出答案。

返回类型不同

上面说了VS会根据你使用的数据集合的类型来自动使用相应命名空间下的方法，这是因为VS选择了最优的使用。当返回数据集合如Where方法时，Enumerable的Where返回的IEnumerable<>，Queryable的Where返回的是IQueryable<>,这两个集合有什么区别呢？IEnumerable<>是本地集合存在内存中，而IQueryable<>属于离线集合，在使用到返回的集合IQueryable<>才加载数据，属于延迟加载

```
Model1Container db = new Model1Container();
IQueryable<User> userSet = db.UserSet.Where(u => u.Id > 0);
```

上面的 db.UserSet使用了Queryable下的Where方法，这个方法需要操作数据库，但是不立刻操作数据库，而是等用到userSet时才从数据库取出数据，属于延迟加载，提高了效率。

传递参数不同
还是这一段代码：

```
int[] y = { 0, 9, 2, 3, 5 };
Model1Container db = new Model1Container();
// 使用Enumerable下的方法
var x2 = x.Where(u => u > 0);
// 使用Queryable下的方法
var userSet = db.UserSet.Where(u => u.Id > 0);
```

当我们封装一些方法时需要传递参数，如上面的代码，我们想要将

```
u => u.Id > 0
```

用参数代替。
别看都是u => u.Id > 0，但是是两个不同的类型，一个是Func,一个是 Expression,如下：

```
Expression<Func<User, bool>> expression = u => u.Id > 0;
Func<int, bool> expression2 = u => u> 0;

IEnumerable<int> x2 = x.Where(expression2);
IQueryable<User> userSet = db.UserSet.Where(expression);
```

Func和Expression将单独讲解，这里只需要知道，可以将 lambda 表达式 分配到Func<> 委托中当作参数传递；而Expression<>是将Func<>表示成数据结构。
对Func有兴趣的可以看下C#中的Func<>

总结
Enumerable在本地集合中使用，直接加载到内存中，Queryable在操作数据库时延迟加载。
Enumerable需要的参数是委托类型Func<>，Queryable需要的参数类型是数据结构。