عنوان: **استفاده از OfType یا Cast در Linq**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۴/۲۸ ۲۰:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: LINQ, Generics, Casting, Collections

تقریبا تمام توسعه دهندگان دات نت با تکنولوژی Linq و Lambda Expressionها آشنایی دارند. همان طور که می‌دانیم Extension Methodهای موجود در فضای نام System.Linq فقط بر روی مجموعه ای از داده‌ها که اینترفیس IEnumerable<t> که در فضای نام System.Collections.Generic قرار دارد را پیاده سازی کرده باشند قابل اجرا هستند. مجموعه داده‌های جنریک فقط قابلیت نگهداری از یک نوع داده که به عنوان پارامتر T برای این مجموعه تعریف می‌شود را داراست.
نکته: البته در مجموعه هایی نظیر Dictionary یا سایر Collectionها امکان تعریف چند نوع داده به عنوان پارامتر وجود دارد. نکته مهم این است که داده‌های استفاده شده در این مجموعه ها، حتما باید از نوع پارامتر تعریف شده باشند.
اگر در یک مجموعه داده قصد داشته باشیم که داده هایی با نوع مختلف را ذخیره کنیم و در جای مناسب آن‌ها را بازیابی کرده و در برنامه استفاده نماییم چه باید کرد. به عنوان یک پیشنهاد می‌توان از مجموعه‌های موجود در فضای نام System.Collection بهره بگیریم. اما همان طور که واضح است این مجموعه از داده‌ها به صورت جنریک نمی‌باشند و امکان استفاده از Queryهای Linq در آن‌ها به صورت معمول امکان پذیر نیست. برای حل این مشکل در دات نت دو متد تعبیه شده است که وظیفه آن تبدیل این مجموعه از داده‌ها به مجموعه ای است که بتوان بر روی آن‌ها Queryهای از جنس Linq یا Lambda Expression را اجرا کرد.
Cast
OfType

#مثال 1
فرض کنید یک مجموعه مثل زیر داریم:

```
ArrayList myList = new ArrayList();

  myList.Add( "Value1" );
  myList.Add( "Value2" );
  myList.Add( "Value3" );

  var myCollection = myList.Cast<string>();
```

در مثال بالا یک Collection از نوع ArrayList ایجاد کردیم که در فضای نام System.Collection قرار دارد. شما در این مجموعه می‌توانید از هر نوع داده ای که مد نظرتان است استفاده کنید. با استفاده از اپراتور Cast توانستیم این مجموعه را به نوع مورد نظر خودمان تبدیل کنیم و در نهایت به یک مجموعه از IEnumerable<T> برسیم. حال امکان استفاده از تمام متدهای Linq امکان پذیر است.
#مثال دوم:

```
ArrayList myList = new ArrayList();

    myList.Add( "Value1" );
    myList.Add( 10 );
    myList.Add( 10.2 );

   var myCollection = myList.Cast<string>();
```

در مثال بالا در خط آخر با یک runtime Error مواجه خواهیم شد. دلیلش هم این است که ما از در ArrayList خود داده‌های غیر از string نظیر int یا double داریم. درنتیجه هنگام تبدیل داده‌های int یا double به string یک Exception رخ خواهد داد. در این گونه موارد که در لیست مورد نظر داده‌های غیر هم نوع وجود دارد باید متد OfType را جایگزین کنیم.

```
ArrayList myList = new ArrayList();

myList.Add( "Value1" );
myList.Add( 10 );
myList.Add( 10.2 );
```

```
var doubleNumber = myList.OfType<double>().Single();
var integerNumber = myList.OfType<int>().Single();
var stringValue = myList.OfType<string>().Single();
```

تفاوت بین متد Cast و OfType در این است که متد Cast سعی دارد تمام داده‌های موجود در مجموعه را به نوع مورد نظر تبدیل کند ولی متد OfType فقط داده‌های از نوع مشخص شده را برگشت خواهد داد. حتی اگر هیچ آیتمی از نوع مورد نظر در این مجموعه نباشد یک مجموعه بدون هیچ داده ای برگشت داده می‌شود.