禪

دسترسی به منابع دادهای با LINQ

تکنیکهای کاربردی استخراج دادهها با لینک (بخش اول)

.NET Language-Integrated Query (LINQ)

C#

VB

Others…

.NET Language-Integrated Query

LINQ enabled data sources

LINQ enabled ADO.NET

LINQ To Objects

LINQ To Datasets

LINQ To SQL

LINQ To Entities

LINQ To XML

<book>
<title/>
<author/>
<price/>
</book>

Objects

Relational

XML

لینک، LINQ سرنام Language-Integrated Query، از جمله راهکارهای مؤثر در دسترسی به دادهها است. امروزه، رقابت اصلی میان تولیدکنندگان ابزارهای برنامهنویسی بر سر دسترسی سریع و آسان به منابع دادهای است. لینک چارچوب مستحکمی را به وجود میآورد که دسترسی به اطلاعات ذخیرهشده روی طیف گستردهای از منابع دادهای را امکانپذیر میکند و همانند پلی برای برقراری ارتباط میان اشیا برنامهنویسی و دادهها استفاده میشود.

پیش از ظهور لینک، محاورهها با رشتههای سادهای که در یک برنامه کاربردی ساخته میشدند و حتی فاقد ویژگی IntelliSense بودند، استفاده میشدند. اما با ظهور لینک، این شکل دسترسی به دادهها کاملاً متحول شد و برنامهنویسان توانستند با حداقل کدنویسی محاورهها را از یک رشته ساده خارج کرده و به محاورههایی تبدیل کنند که مدیریت بیشتر روی آنها امکانپذیر باشد. منعطف بودن لینک باعث محبوبیت آن شد. کدنویسان بدون اینکه درگیر قواعد بسیار پیچیده دسترسی به منابع دادهای مختلف شوند، موفق شدند بهراحتی برای دسترسی به دادههایی که روی منابع مختلفی همچون بانکهای اطلاعاتی رابطهای، اسناد xml و... قرار دارند، از لینک استفاده کنند. با این مقدمه، به سراغ معرفی تعدادی از تکنیکهای پرکاربرد در لینک خواهیم رفت که برای دسترسی به دادههایی که روی منابع مختلف قرار دارند، استفاده میشود. اما ابتدا باید با ساختمان درونی محاورهها در لینک آشنا شویم.

محاوره چیست؟

پرسوجو (Query) که به نام محاوره نیز شناخته میشود، رشتهای متنی است که بهمنظور بازیابی دادهها از یک منبع دادهای، بهروزرسانی، اضافه کردن و حذف دادهها، از آن استفاده میشود . لینک با هدف ارائه یک مدل ساده از محاورهها در اختیار برنامهنویسان قرار گرفت؛ مدلی که قادر است به منابع دادهای مختلف به شیوه کدنویسی یا ویزاردی متصل شود. برای این منظور مجموعهای از متدهای اصلی که عملگرهای استاندارد محاوره (SQO) نام دارند، لینک را یاری میکنند . طیف گستردهای از این متدها به شکل ترتیبی کار میکنند؛ به این معنی که شی مورد محاوره از نوع یکی از رابطهای IEnumerable<T> یا IQueryable<T> خواهد بود . شایان ذکر است IQueryable که برای محاورههای LINQ TO SQL استفاده میشود، در نهایت به IEnumerable تبدیل میشود. در نتیجه یک محاوره لینک همواره با اشیا سروکار دارد .

مطلب پیشنهادی

Bloks زیرساختی آموزش محور

گوگل در حال توسعه پلتفرم قدرتمندی ویژه آموزش به کودکان است

نکته بارز و شاخص این مدل به ویژگی یکدست بودن آن بازمیگردد؛ به طوری که برنامهنویسان در اکثر مواقع از الگوی ثابتی بهمنظور بازیابی دادهها از منابع دادهای استفاده میکنند . محاورههای ساختهشده توسط لینک، عبارات باقاعدهای هستند که همراه با فیلترهای مختلفی همچون مرتبسازی، تجمع و... قابل بهرهبرداری هستند . استخراج ساختمند دادهها در عمل به طراحان و بهویژه طراحان وب کمک فراوانی میکند . روشهای SQO روشهای توسعهیافتهای هستند که از کلاسهای Enumerable و Queryable تعریف میشوند. بعضی منابع این روشها را عملگرهای پرسوجو نامگذاری کردهاند . همگی این روشها در فضایی به نام System.Linq قرار دارند . بهطور کلی محاورهها در لینک، در پنج گروهی که در جدول شماره یک مشاهده میکنید، قرار میگیرند .

LINQ to Object	به برنامه‌نویس اجازه می‌دهد محاوره‌ها را روی ساختارهای داده‌ای مقیم در حافظه همانند فهرست‌های ژنریک و آرایه‌ها استفاده کند.
LINQ to Dataset	ساخت محاوره‌ها برای کار با مجموعه داده‌ها در Ado.Net را امکان‌پذیر می‌سازد.
LINQ to SQL	به‌منظور پیاده‌سازی محاوره‌ها برای استخراج و به‌روزرسانی داده‌ها روی بانک‌های اطلاعاتی SQL Server استفاده می‌شود.
LINQ to Entities	این گروه به برنامه‌نویسان اجازه می‌دهد محاوره‌ها را برای دسترسی به بانک‌های اطلاعاتی مختلف طراحی کنند.
LINQ to XML	امکان طراحی محاوره‌ها و ویرایش داده‌های XML مقیم در حافظه یا داده‌های XML ذخیره‌شده در یک فایل XML را فراهم می‌آورد.

[illegible] *1:* [illegible].

[illegible].
LINQ to Active Directory [illegible] LINQ to [illegible]. [illegible] Objects [illegible] LINQ to SQL [illegible]. [illegible] LINQ [illegible] SQL [illegible] to SQL [illegible].

From	اشاره به منبع داده‌ای دارد که محاوره روی آن اجرا می‌شود و داده‌ها را از آن استخراج می‌کند.
Where	به شرطی اشاره دارد که داده‌ها بر مبنای آن از منبع داده‌ای استخراج می‌شوند.
Orderby	نحوه سازمان‌دهی و مرتب‌سازی داده‌های بازیافت‌شده را مشخص می‌کند.
Select	محتوای داده‌های بازیافت‌شده را مشخص می‌کند.
Join	این عملگر همانند inner join در اس‌کیوال است و دو محاوره را با یکدیگر ترکیب می‌کند.
Group	مقادیر برگشتی را گروه‌بندی می‌کند.

[illegible] 2: [illegible].

[illegible]

[illegible] foreach [illegible].

```
static void Main(string[] args)
      {

       int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };
      var myQuery =
         from num in numbers
         where (num % 2) == 0
         select num;
      foreach (int num in myQuery)
      {
         Console.Write("{0,1} ", num);
      }
     }
```

[illegible]

[illegible] 1 [illegible].

1 תמונה: תהליך ביצוע שאילתה של שאילתת LINQ

שאילתות LINQ מבוססות

שאילתות LINQ מבוססות על הממשק IEnumerable<T>. כאשר מהדר יוצר שאילתה מבוססת, הוא יוצר מחלקה שמממשת את IEnumerable<T> או IEnumerable שאפשר לעבור עליה בעזרת הלולאה foreach. בזמן הריצה, הלולאה foreach משתמשת בשיטות של IEnumerable<T> או IEnumerable כדי לבצע את השאילתה ולהחזיר את התוצאות. שאילתות LINQ מבוססות IEnumerable<T> נקראות לעיתים שאילתות בזיכרון, מכיוון שהן עובדות ישירות על אובייקטים בזיכרון. שאילתות LINQ מבוססות IQueryable<T>, לעומת זאת, יוצרות עצי ביטוי שמהדר או ספריה אחרת יכולים לבדוק בזמן הריצה כדי לקבוע את הדרך הטובה ביותר לגשת למקור נתונים חיצוני ולהריץ את השאילתה. לדוגמה, LINQ to SQL מתרגם שאילתות LINQ לשאילתות SQL שמועברות לשרת SQL Server לביצוע. LINQ to Dataset מאפשר שימוש בשאילתות LINQ כנגד מערכי נתונים של ADO.NET. LINQ to XML מאפשר שימוש בשאילתות LINQ כנגד מסמכי XML.

שאילתות LINQ מאפשרות לבצע פעולות מורכבות על נתונים בקלות יחסית

בדוגמה הבאה, נוצרת שאילתה שמחזירה את כל המחרוזות במערך שאורכן הוא חמישה תווים. ניתן לראות שהשאילתה משתמשת בתכונה Length של כל מחרוזת, אך ניתן לבצע גם פעולות מורכבות יותר, כגון חיפוש התאמות לביטויים רגולריים או שינוי של מחרוזות המקור.

```
static void Main(string[] args) { String[] myArray = { “One”, “Two”, “Three”, “Four”, “Five” };
        var MyQuery =
            from mystring
```

```
            in myArray
         select mystring;
      foreach(var str in MyQuery)
      Console.WriteLine(str); }
```

فهرست شماره دو

محاوره شماره دو؛ محدود کردن دادههای دریافتی از یک منبع داده

محاوره شماره یک، همه دادههای درون یک آرایه را بازمیگرداند؛ اما اگر در نظر داشته باشیم تنها دادههای خاص خود را استخراج کنیم، باید از کلمه کلیدی where استفاده کنیم . این کلمه کلیدی به شما اجازه میدهد تا شرطی را روی یک محاوره پیادهسازی کنید . در اغلب موارد، محاورهها به فیلتر نیاز دارند؛ به دلیل اینکه در بیشتر زمانها نیازی نداریم کل مجموعه دادهها را از درون یک منبع دادهای استخراج کنیم؛ بهویژه زمانی که منبع داده هدف ما یک بانک اطلاعاتی مشتمل بر دهها هزار رکورد باشد . مکان قرارگیری کلمه where در یک محاوره، بعد از کلمه from و قبل از کلمه کلیدی select است. در فهرست شماره سه از ترکیب نحوی where بهمنظور محدودکردن خروجی دادههایی که اندازه آنها از سه کاراکتر بیشتر است، استفاده کردهایم .

```
   static void Main(string[] args)
   {

     String[] myArray = { “One”, “Two”, “Three”, “Four”, “Five” };
       var MyQuery =
from mystring
in myArray
where mystring.Length > 3
select mystring;

       foreach(var str in MyQuery)
       Console.WriteLine(str);

   }
```

فهرست شماره سه

محاوره شماره سه؛ مرتبسازی و گروهبندی دادههای بازیافتشده در یک محاوره

برای چینش دادههای بازیافتی در یک محاوره از کلمه کلیدی orderby استفاده میشود . با استفاده از این کلمه کلیدی نحوه مرتبسازی دادهها، مطابق با نیاز کاری شما خواهند بود . در کنار عملگر orderby روشها و عملگرهای دیگری نیز برای مرتبسازی دادهها در اختیار شما قرار دارند . فهرست شماره چهار نحوه مرتبسازی دادهها با عملگر orderby را نشان میدهد .

```
 class Program
 {
   public class myclass
   {
   public string Name { get; set; }
   public int Age { get; set; }
 }
   static void Main(string[] args)
   {

     myclass[] mystrings = { new myclass { Name=”Barley”, Age=8 },
         new myclass { Name=”Boots”, Age=4 },
```

```
            new myclass { Name=”Whiskers”, Age=1 } };

        IEnumerable<myclass> query = mystrings.OrderBy(Program => Program.Age);

        foreach (myclass mystr in query)
        {
           Console.WriteLine(“{0} - {1}”, mystr.Name, mystr.Age);
        }

    }
}
```

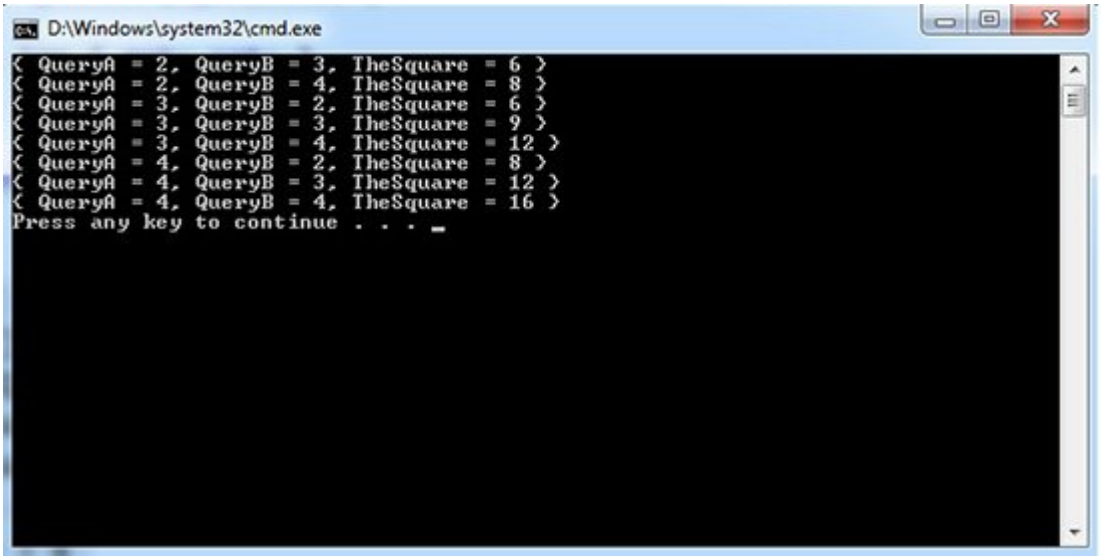
[illegible]

[illegible]

```
static void Main(string[] args)
    {

        Int32[] FirstArray= {1,2,3,4,5};
        Int32[] SecondArray = { 6,7,8,5,4};

        var MyQuery = from QueryA in FirstArray
                      from QueryB in SecondArray
                       where QueryA == QueryB
                       select new { QueryA, QueryB };

        foreach(var str in MyQuery)
        Console.WriteLine(str);
    }
```

[illegible]

[illegible]



D:\Windows\system32\cmd.exe

```
{ QueryA = 2, QueryB = 3, TheSquare = 6 }
{ QueryA = 2, QueryB = 4, TheSquare = 8 }
{ QueryA = 3, QueryB = 2, TheSquare = 6 }
{ QueryA = 3, QueryB = 3, TheSquare = 9 }
{ QueryA = 3, QueryB = 4, TheSquare = 12 }
{ QueryA = 4, QueryB = 2, TheSquare = 8 }
{ QueryA = 4, QueryB = 3, TheSquare = 12 }
{ QueryA = 4, QueryB = 4, TheSquare = 16 }
Press any key to continue . . .
```

شکل 2: نحوه پیادهسازی یک شرط روی محاوره

```
static void Main(string[] args)
    {

       Int32[] ArrayA = {1,2,3,4};
        Int32[] ArrayB = { 1,2,3,4};
        var MyQuery =
from QueryA in ArrayA
from QueryB in ArrayB
let TheSquare = QueryA * QueryB
where TheSquare > 4
select new { QueryA, QueryB, TheSquare };
        foreach(var str in MyQuery)
        Console.WriteLine(str); }
```

فهرست شماره پنج

در شماره آینده به بررسی تکنیکهای دیگر مربوط به لینک خواهیم پرداخت .

نویسنده :
حمیدرضا_تائبی
دسته بندی:
کارگاه
تاریخ انتشار :
10:05 - 24/11/1395
برچسب:
LINQ - برنامهنویسی

□□□□□ منبع: https://www.shabakeh-mag.com/workshop/6506