مروری بر تغییرات از VB6 به VB.NET

مقدمه

با عرضه پلاتفرم .NET از طرف مایکروسافت ویژوال بیسیک به صورت یک زبان کاملا شی گرا ظهور کرد. در این مقاله سعی کرده ام لیست تغییرات زبان ویژوال بیسیک را با مثالهایی در هر قسمت بیان کنم. لیست حاضر در اینجا کامل نیست اما بسیاری از مباحث اساسی را پوشش داده است. اگر شما یک برنامه نویس VB هستید و می خواهید به سمت VB.NET بروید بهتر است این مقاله را بخوانید.

تغییرات نوع داده:

پلاتفرم .NET سیستم نوع اصلی (Common Type System) را برای همه زبانها فراهم کرده است. این بدین معنی است که همه زبانها باید از انواع حمایت شده توسط زبان اصلی زمان اجرا پشتیبانی کنند. این روش ناسازگاری انواع داده بین زبانهای مختلف را حل می کند. برای مثال در پلاتفرم Win32 نوع داده Integer (عدد صحیح) در زبانهایی مانند C++ ۴ بایت می گیرد در حالی که در VB ۲ بایت. لیستی که در زیر مشاهده می کنید تغییرات اساسی هستند که در مورد انواع داده در VB.NET وجود دارد:

- در .NET نوع داده Integer همیشه ۴ بایتی است.
- VB.NET نوع داده Currency ندارد. به جای آن می توانید نوع داده Decimal را جایگزین کنید.
- در VB.NET یک نوع داده جدید به نام Char قرار داده شده است. نوع داده Char ۲ بایت می گیرد و می تواند یک کاراکتر یونیکد را ذخیره کند.
- نوع داده Variant در VB.NET وجود ندارد. برای دسترسی به نتیجه ای مانند نوع داده Variant می توانید از نوع داده Object استفاده کنید. (یک متغیر از نوع داده Object می تواند به هر نوع داده ای اشاره داشته باشد.)
- در VB.NET چیزی به اسم رشته های با طول ثابت وجود ندارد.
- در VB6 ما از کلمه کلیدی Type برای تعریف ساختار دادذه تعریف شده توسط کاربر استفاده می کردیم. VB.NET لغت کلیدی Structure را برای همان عمل در اختیار قرار داده است. Syntax مورد استفاده هم همان است. به این صورت که:

```
Structure MyStructure1
...
...
End Structure
```

اعلان متغیرها:

این مثال ساده را در VB6 در نظر بگیرید:

```
Dim x,y As Integer
```

در این مثال VB6 متغیر x را از نوع Variant و متغیر y را از نوع Integer در نظر گرفته است که آن چیزی نیست که ما انتظار داشتیم. VB.NET این مشکل را برطرف کرده است ساختن xوy هر دو از نوع Integer. همچنین VB.NET به شما اجازه می دهد تا مقادیر اولیه را به متغیرها در همان دستور اعلان اختصاص دهید:

```
Dim Str1 As String=”Hello”
```

همچنین VB.NET متغیرهای فقط خواندنی تولید می کند. متغیرهای فقط خواندنی مثل ثابتها نیستند و می توانند بدون مقدار اولیه اعلان شوند اما وقتی مقداری به آن تخصیص دادید دیگر نمی توان آن را تغییر داد. مثال:

```
Dim ReadOnly x As Integer
```

در جای دیگری از کد دستور زیر را می نویسیم:

```
X=100
```

حالا x نمی تواند تغییر داده شود و دستور زیر خطایی را ایجاد می کند:

```
X=200
```

آرایه ها:

در VB6 شما می توانستید در برنامه نویسی حد پایین و حد بالای آرایه را تعیین کنید. در VB.NET حد پایین یک آرایه همیشه صفر است. وقتی یک آرایه مانند زیر تعریف می کنید:

```
Dim aStates(50) As String
```

در حقیقت ۵۱ عنصر ساخته شده است با حد پایین صفر و حد بالای ۵۰ (توجه کنید که در نسخه بتا از کمپایلر VB.NET دستور فوق ۵۰ عنصر می سازد با حد پایین صفر و حد بالای ۴۹).

میدان دید متغیر:
کد VB6 زیر را ملاحظه کنید:

```
If x=y then
  Dim z as integer
  ' other code
End If

z=100
```

کد بالا بصورت کامل در VB6 اجرا مي شود زيرا در اينجا محدوده ديد سطح بلاك وجود ندارد. (محدوده سطح بلاك در زبانهاي برنامه نويسي پيشرفته در نظر گرفته شده است مانند C++). متغير تعريف شده درون بلاك مانند متغير تعريف شده در بلاك If…Then وقتي جمله هاي درون بلاك به پايان مي رسد از محدوده ديد بيرون مي افتد. بنابراين دستيابي به متغير Z كه خارج از بلاك If…Then تعريف شده است باعث رخ دادن يك خطا در زبانهاي برنامه نويسي پيشرفته خواهد شد. VB.NET متمايل به VB6 نيست و داراي ميدان ديد متغير از نوع سطح بلاك مي باشد.

عبارات Set و Let:
در زبان VB6 از عبارت Set براي انتصاب يك نمونه شي به يك متغير استفاده مي شود. اين كار به خاطر وجود خواص پيش فرض، لازم بود. براي اينكه به VB6 بگوييم كه عمل انتصاب يك متغير به خود شي را مي خواهيم انجام دهيم (در مقابل مقدار دادن به خاصيت پيش فرض شي) شما بايد از لغت كليدي Set استفاده كنيد. براي مثال در VB6 :

```
Dim x As Variant
Dim strName As String
```

عمل انتصاب مقدار Text1.Text به متغير strName (خاصيت Text خاصيت پيش فرض كنترل TextBox در VB6 است) را خط زير انجام مي دهد:

```
strName=Text1
```

در ادامه ما واقعا شي TextBox را به متغير x انتصاب داده ايم. به اين نكته توجه كنيد كه ما از لغت كليدي Set استفاده كرده ايم تا VB6 بفهمد كه مي خواهيم خود شي را به X انتصاب دهيم نه خاصيت پيش فرض آن را:

```
Set x=Text1
```

در VB.NET لغت كليدي Set لازم نيست. لغت كليدي Let نيز از Syntax زبان VB.NET حذف شده است.

بدام انداختن خطا (Error Handling):
سرانجام بدام اندازي خطاي ساخت يافته به ويژوال بيسيك اضافه شد. لغات كليدي Try,Catch,Finally بدام اندازي خطا را آسان كرده اند و مي توان VB.NET را در زمره زبانهايي مانند C++ يا C# به حساب آورد. مدل Try…Catch به توسعه دهندگان اجازه مي دهد كدي را كه ممكن است باعث استثنا (يا خطا) شود را در يك بلاك Try قرار دهند. اگر كد مورد نظر يك استثنا را باعث شود (مترادف رخ دادن يك خطا) كدي كه در بلاك Catch قرار دارد اجرا مي شود؛ كدي كه در اين بلاك قرار دارد بايد براي بدام اندازي استثنا باشد.

توجه كنيد كه تكنيكهاي قديمي بدام اندازي خطا متعلق به VB6 (مانند On Error Resume Next و غيره) براي سازگاري با نسخه هاي قبلي هنوز توسط VB.NET پشتيباني مي شوند؛ هرچند وقتي شما برنامه جديدي را در VB.NET مي نويسيد، بايد شجاعانه كوشش كنيد تا از تكنيكهاي قديمي استفاده نكنيد.

قطعه كدهايي كه مشاهده مي كنيد تكنيكهاي مختلف بدام اندازي خطا را در VB.NET شرح مي دهد:

```
Try
…
Catch
…
End Try
```

كد بالا بسادگي هر استثناي رخداده شده در كدي كه در بلاك Try قرار دارد را بدام مي اندازد. VB.NET به شما اجازه مي دهد تا استثناهاي خاص را با استفاده از چندين بلاك Catch بدام بيندازيد:

```
Try
…
Catch e1 As NullPointerExeption
…
Catch e2 As Exeption
…
End Try
```

همچنين براي گرفتن استثناهاي دوباره تعريف شده مي توانيد كلاسهاي سفارشي خودتان كه از كلاس پايه System.Exeption به ارث مي برند را بسازيد. شما مي توانيد از Trow براي استثناهاي خودتان استفاده كنيد (ساده تر از استفاده از متد Raise از شي Err در VB است):

```
If myVar<1000 Then
        Throw new Exeption(“Bussiness logic Error”)
End If
```

متدهاي استاتيك:

اكنون VB.NET به شما اجازه مي دهد كه متدهاي استاتيك براي كلاسها بسازيد. متدهاي استاتيك متدهايي هستند كه برنامه نويس مي تواند آنها را بدون نياز به ساختن نمونه اي از كلاس فراخواني كند. براي مثال، اگر كلاسي به نام Foo با يك متد غير استاتيك با عنوان NonStatic() و يك متد استاتيك بنام Static() داشته باشيد، بايد متد استاتيك را مانند كد زير فراخواني كنيد:

```
Foo.Static()
```

همچنين براي فراخواني متد غير استاتيك بايد يك نمونه از كلاس ساخته شود مانند زير:

```
Dim objFoo As New Foo()
ObjFoo.NonStatic()
```

براي ساختن متد استاتيك در يك كلاس VB.NET بسادگي بعنوان پيشوند تعريف متد از لغت كليدي Shared استفاده كنيد.

روالها و توابع:

در VB6 همه پارامترهاي روالها بطور پيش فرض بصورت ارسال با مرجع (ByRef) به روالها ارسال مي شوند. در VB.NET پارامترها بطور پيش فرض بصورت ارسال با مقدار (ByVal) به روالها ارسال مي شوند. در فراخواني روالها و توابع چه پارامتر داشته باشند چه نداشته باشند بايد پرانتز داشته باشند. در VB6 توابع با استفاده از Syntax زير مقادير را برگشت مي دهند:

```
FunctionName=Return_Value
```

در VB.NET شـما مـي توانـيد از لغـت كلـيدي Return (Return_Value Return) بـراي برگـرداندن مقاديـر اسـتفاده كنـيد، همچنـين مـي توانـيد از Syntax قديمـي برگشـت مقادير نيز استفاده كنيد.

Syntax خواص:

در VB6 براي ساختن خواص در كلاسها از Proprty Get و Property Set/Let استفاده مي كرديم. دو روتين مجزاي زير را مشاهده كنيد:

```
Public Property Get PropertyName() As DataType
…
End Property

Public Property Let PropertyName(Value As DataType)
…
End Property
```

در VB.NET نحوه ساخت خواص اندكي تغيير كرده است. بجاي داشتن دو روتين مجزاي Property Get و Property Let/Set در اينجا آن دو را در يك عبارت Property ادغام كرده اند.

```
Public  [ReadOnly/WriteOnly] Property PropertyName As DataType
Get
        Return m_Var
End Get

Set
        m_Var=Value
End Set
End Property
```

نتيجه گيري:

در اينجا بعضي از تغييراتي كه در Syntax و Semantic زبان VB.NET اتفاق افتاده ، آمده است اما در اينجا بيشتر تغييراتي ذكر شده است كه براي توسعه دهندگان ASP.NET اهميت بيشتري دارند.

چيزهايي كه براي كار كردن با VB.NET در ساخت صفحات ASP.NET براي شما اهميت بيشتري دارند عبارتند از:

۱- متغيرها بايد نوع داشته باشند و متغيرهاي بدون نوع از نوع Variant محسوب نمي شوند. بعنوان مثال اگر شما به يك متغير Integer نياز داشته باشيد بايد بجاي عبارت Dim I از عبارت Dim I As Integer استفاده كنيد.

۲- فراموش نكنيد كه در فراخواني روالها و توابع در VB.NET بايد پارامترها درون پرانتز قرار گيرند. بعنوان مثال عبارت زير يك خطا توليد خواهد كرد:

```
Response.Write “Hello,World!”
```

بجاي آن بايد پارامترها را درون پرانتزها قرار دهيد:

```
Response.Write(“Hello,World!”)
```

۳- VB.NET خواص پيش فرض را پشتيباني نمي كند.

٤- وقتي آرايه ها را تعريف مي كنيد دقت داشته باشيد. همه آرايه ها در VB.NET داراي حد پايين صفر و حد بالايي است كه شما براي آن تعريف كرده ايد.

برنامه نويسي خوبي داشته باشيد!

نویسنده: **Bipin Joshi**
مترجم: **جادوگر ویژوال بیسیک**
وبلاگ: **http://vblog.persianblog.com**