

عنوان: ماژول ها و فضای نام(namespace)

نویسنده: مسعود پاکدل

تاریخ: ۱۳۹۲/۰۳/۲۱ ۱:۳۶

آدرس: www.dotnettips.info

برچسب‌ها: F#, Programming

در F# ماژول به گروهی از کدها، توابع، انواع داده‌ها و شناسه‌ها گفته می‌شود و کاربرد اصلی آن برای قرارگیری کدها مرتبط به هم در یک فایل است و هم چنین از تناقص نام‌ها جلوگیری می‌کند. در F# در صورتی که توسط برنامه نویس ماژول تعریف نشود هر source file یک ماژول در نظر گرفته می‌شود. برای مثال:

```
// In the file program.fs.  
let x = 40
```

بعد از کامپایل تبدیل به کد زیر می‌شود.

```
module Program  
let x = 40
```

هم چنین امکان تعریف چند ماژول در یک source file نیز میسر است. به این صورت که باید برای هر ماژول محلی یک نام اختصاص دهید. در مثال بعدی دو تا ماژول را در یک فایل به نام mySourceFile قرار می‌دهیم.

```
module MyModule1 =  
    let module1Value = 100  
  
    let module1Function x =  
        x + 10  
  
// MyModule2  
module MyModule2 =  
  
    let module2Value = 121  
  
    let module2Function x =  
        x * (MyModule1.module1Function module2Value)
```

در آخرین خط همان طور که مشاهده می‌کنید با استفاده از نام ماژول می‌توانیم به تعاریف موجود در ماژول دسترسی داشته باشیم. (MyModule1.module1Function).

استفاده از یک ماژول در فایل‌های دیگر.

گاهی اوقات نیاز به استفاده از تعاریف و توابع موجود در ماژولی داریم که در یک فایل دیگر قرار دارد. در این حالت باید به روش زیر عمل کنیم.

فرض بر این است ماژول زیر در یک فایل به نام ArithmeticFile قرار دارد.

```
module Arithmetic  
  
let add x y =  
    x + y  
  
let sub x y =  
    x - y
```

حال قصد استفاده از توابع بالا رو در یک فایل و ماژول دیگر داریم.
#1 روش اول (دقیقا مشابه روش قبل از نام ماژول استفاده می‌کنیم)

```
let result1 = Arithmetic.add 5 9
```

#2 روش دوم (استفاده از open)

```
open Arithmetic
let result2 = add 5 9
```

ماژول های تودرتو

در F# می توانیم یک ماژول را درون ماژول دیگر تعریف کنیم یا به عبارت دیگر می توانیم ماژولی داشته باشیم که خود شامل چند تا ماژول دیگر باشد. مانند:

```
module Y =
    let x = 1

    module Z =
        let z = 5
```

روش تعریف ماژول های تودرتو در F# در نگاه اول کمی عجیب به نظر میرسد. جداسازی ماژول های تودرتو به وسیله دندانه گذاری یا تورفتگی انجام می شود. ماژول Z در مثال بالا به اندازه چهار فضای خالی جلوتر نسبت به ماژول Y قرار دارد در نتیجه به عنوان ماژول داخلی Y معرفی می شود.

```
module Y =
    let x = 1

module Z =
    let z = 5
```

در مثال بالا به دلیل اینکه ماژول Z و Y از نظر فضای خالی در یک ردیف قرار دارند در نتیجه ماژول تودرتو نیستند. حال به مثال بعدی توجه کنید.

```
module Y =
module Z =
    let z = 5
```

در این مثال ماژول X به عنوان ماژول داخلی Y حساب می شود. دلیلش هم این است که ماژول Y بدنه ندارد در نتیجه ماژول Z بلافاصله بعد از آن قرار می گیرد که کامپایلر اونو به عنوان ماژول داخلی حساب می کنه. اما برای اینکه مطمئن شود که قصد شما تولید ماژول تودرتو بود یک Warning میدهد. برای اینکه Warning رو مشاهده نکنیم می تونیم کد بالا رو به صورت زیر بازنویسی کنیم:

```
module Y =
    module Z =
        let z = 5
```

فضای نام (namespace)

مفهوم فضای نام کاملاً مشابه مفهوم فضای نام در C# است و راهی است برای کپسوله سازی کدها در برنامه. مفهوم namespace با مفهوم module کمی متفاوت است.

ساختار کلی

```
namespace [parent-namespaces.]identifier
```

چند نکته درباره namespace

1# اگر قصد داشته باشید که از فضای نام در کدهای خود استفاده کنید باید اولین تعریف در source file برنامه تعریف namespace باشد.

2# امکان تعریف شناسه یا تابع به صورت مستقیم در namespace وجود ندارد بلکه این تعاریف باید در ماژول ها یا type ها نظیر تعریف کلاس قرار گیرند.

#3 امکان تعریف فضای نام با استفاده از تعاریف ماژول نیز وجود دارد(در ادامه به بررسی یک مثال در این زمینه می پردازیم)

تعریف *namespace* به صورت مستقیم:

```
namespace Model
type Car =
    member this.Name = "BMW"
module SetCarName =
    let CarName = "Pride"
```

تعریف *namespace* به صورت غیر مستقیم (استفاده از module)

```
module Model.Car
module SetCarName =
    let CarName = "Pride"
```

فضای نام های تودرتو

همانند ماژول ها امکان تعریف فضای نام تودرتو نیز وجود دارد. یک مثال در این زمینه:

```
namespace Outer
    type OuterMyClass() =
        member this.X(x) = x + 1
namespace Outer.Inner
    type InnerMyClass() =
        member this.Prop1 = "X"
```

همانند فضای نام های در C# با استفاده از (.) می توانیم فضای نام های تودرتو ایجاد کنیم. در مثال بالا فضای نام Inner به عنوان فضای نام داخلی Outer تعریف شد است. برای دسترسی به کلاس InnerMyClass باید تمام مسیر فضای نام رو ذکر کنیم.

```
Outer.Inner.InnerMyClass
```