عنوان: **الگوی Matching**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۱۷ ۱۵:۳
آدرس: www.dotnettips.info
برچسب‌ها: F#, Programming

الگوی Matching در واقع همون switch در اکثر زبان‌ها نظیر C# یا C++ است با این تفاوت که بسیار انعطاف پذیرتر و قدرتمندتر است. در برنامه نویسی تابع گرا، هدف اصلی از ایجاد توابع دریافت ورودی و اعمال برخی عملیات مورد نظر بر روی مقادیر با استفاده از تعریف حالات مختلف برای انتخاب عملیات است. الگوی Matching این امکان رو به ما می‌ده که با استفاده از حالات مختلف یک عملیات انتخاب شود و با توجه به ورودی یک سری دستورات رو اجرا کنه. ساختار کلی تعریف آن به شکل زیر است:

```
match expr with
| pat1 -> result1
| pat2 -> result2
| pat3 when expr2 -> result3
| _ -> defaultResult
```

راحت‌ترین روش استفاده از الگوی Matching هنگام کار با مقادیر است. اولین مثال رو هم در فصل قبل در بخش توابع بازگشتی با هم دیدیم.

```
let booleanToString x =
match x with false -> "False"
| _ -> "True"
```

در تابع بالا ورودی ما اگر false باشد "False" و اگر true باشد "True" برگشت داده می‌شود. _ در مثال بالا دقیقا همون default در switch سایر زبان هاست.

```
let stringToBoolean x =
match x with
| "True" | "true" -> true
| "False" | "false" -> false
| _ -> failwith "unexpected input"
```

در این مثال (دقیقا بر عکس مثال بالا) ابتدا یک string دریافت می‌شود اگر برابر "True" یا "true" بود مقدار true برگشت داده میشود و اگر برابر "False" یا "false" بود مقدار false برگشت داده می‌شود در غیر این صورت یک FailureException پرتاب می‌شود. خروجی مثال بالا در حالات مختلف به شکل زیر است:

```
printfn "(booleanToString true) = %s"
(booleanToString true)
printfn "(booleanToString false) = %s"
(booleanToString false)
printfn "(stringToBoolean \"True\") = %b"
(stringToBoolean "True")
printfn "(stringToBoolean \"false\") = %b"
(stringToBoolean "false")
```

خروجی : ("printfn "(stringToBoolean \"Hello\") = %b" (stringToBoolean "Hello

```
(booleanToString true) = True
(booleanToString false) = False
(stringToBoolean "True") = true
(stringToBoolean "false") = false
```

Microsoft.FSharp.Core.FailureException: unexpected input at FSI_0005.stringToBoolean(String x) at
<StartupCode$FSI_0005>.$FSI_0005.main@() هم چنین علاوه بر اینکه امکان استفاده از چند شناسه در این الگو وجود دارد، امکان استفاده از And , Or نیز در این الگو میسر است.

```
let myOr b1 b2 =
match b1, b2 with
| true, _ -> true  //b1 true , b2 true or false
| _, true -> true // b1 true or false , b2 true
| _ -> false


printfn "(myOr true false) = %b" (myOr true false)
printfn "(myOr false false) = %b" (myOr false false)
```

خروجی برای کدهای بالا به صورت زیر است:

```
(myOr true false) = true
(myOr false false) = false
```

استفاده از عبارت و شروط در الگوی Matching

در الگوی Matching اگر در بررسی ورودی الگو با یک مقدار نیاز شما را برطرف نمی‌کند استفاده از فیلترها و شروط مختلف هم مجاز است. برای مثال

```
let sign = function
    | 0 -> 0
    | x when x < 0 -> -1
    | x when x > 0 -> 1
```

مثال بالا برای تعیین علامت هر عدد ورودی به کار می‌رود. -1 برای عدد منفی و 1 برای عدد مثبت و 0 برای عدد 0.

عبارت if ... then ... else

استفاده از if در F# کاملا مشابه به استفاده از if در C# است و نیاز به توضیح ندارد. تنها تفاوت در if else است که در F# به صورت elif نوشته می‌شود.

ساختار کلی

```
if expr then
    expr
elif expr then
    expr
elif expr then
    expr
...
else
    expr
```

برای مثال الگوی Matching پایین رو به صورت if خواهیم نوشت.

```
let result =
match System.DateTime.Now.Second % 2 = 0 with
| true -> "heads"
| false -> "tails"
```

#با استفاده از if

```
let result =
if System.DateTime.Now.Second % 2 = 0 then
box "heads"
else
box false
printfn "%A" result
```

در پایان یک مثال مشترک رو به وسیله دستور swith case در C# و الگوی matching در F# پیاده سازی می‌کنیم.

C#	F#

switch (day) {
case 0: return "Sunday";
case 1: return "Monday";
case 2: return "Tuesday";
case 3: return "Wednesday";
case 4: return "Thursday";
case 5: return "Friday";
case 6: return "Saturday";
default:
throw new
ArgumentException("day");
}
	match day with
	| 0 -> "Sunday"
	| 1 -> "Monday"
	| 2 -> "Tuesday"
	| 3 -> "Wednesday"
	| 4 -> "Thursday"
	| 5 -> "Friday"
	| 6 -> "Saturday"
	| _ ->
	invalidArg "day" "Invalid day"