

Regular Expression Menggunakan Visual Basic .NET

Didik Dwi Prasetyo

didik_rpl@yahoo.com

Lisensi Dokumen:

Copyright © 2006 IlmuKomputer.Com

*Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.Com**.*

Di dalam kehidupan sehari-hari, sering kali kita menggunakan karakter regular expression untuk memudahkan pekerjaan. Sebagai contoh, kita dihadapkan pada sejumlah file-file yang bervariasi ekstensinya, sementara yang kita perlukan adalah file-file dengan ekstensi **.doc**. Memutuskan untuk mencari dan file satu per satu tentulah bukan pekerjaan yang efektif. Selain membutuhkan waktu lama, sangat mungkin sekali file tidak ditemukan, karena terselip di antara file-file lainnya atau kurang ketelitian dikarenakan mata sudah mulai mengantuk.

Untuk memudahkan pencarian, Anda tentu akan berpikir bahwa selain ekstensi **.doc**, maka file-file tersebut bukan target yang diharapkan. Dengan demikian, Anda akan melakukan penyaringan sesuai kriteria yang dimaksud. Tanpa menghabiskan waktu, Anda segera mengetikkan perintah **dir *.doc** (atau **ls *.doc** di Unix), karena pada kasus ini Anda hanya bekerja dengan *command prompt*. Hasil yang didapatkan sangat memuaskan, karena yang kita hadapi sekarang hanya file-file **.doc** saja. Tidak berapa lama kemudian, Anda sudah berhasil mendapatkan file yang dimaksud, dan tanpa harus mengorbankan waktu yang sangat berharga bagi Anda.

Ilustrasi di atas memperlihatkan contoh yang cukup sederhana, bagaimana karakter regular expression bekerja menyelesaikan masalah yang rumit. Untuk implementasi yang lebih lanjut, akan dijelaskan cara memanfaatkan karakter-karakter regular expression dalam melakukan validasi data dengan menggunakan Visual Basic .NET.

Regular Expression di Visual Basic .NET

Regular expression atau biasa disingkat regex, merupakan suatu notasi fleksibel dan ringkas untuk menemukan dan menggantikan pola teks. Notasi regular expression terdiri dari dua jenis karakter dasar, yaitu karakter teks literal (normal) dan metakarakter. Karakter normal menyatakan bahwa teks harus eksis di string target, sedangkan metakarakter menyatakan teks dapat bermacam-macam di string target. Regular expression memungkinkan kita menguraikan sejumlah teks guna menemukan pola karakter spesifik. Selain itu, Anda juga bisa mengganti, memodifikasi, atau menghapus suatu substring dengan cepat dan akurat, sesuai kriteria yang kita inginkan.

Hampir semua bahasa pemrograman mengimplementasikan regular expression, begitu pula halnya dengan Visual Basic .NET. Di .NET Framework disediakan namespace `System.Text.RegularExpressions` yang berisi delapan kelas untuk mendukung penggunaan regular expression. Namespace ini menyediakan fungsionalitas yang dapat digunakan pada berbagai platform atau bahasa yang berjalan di .NET Framework,

termasuk C#, C++, dan J#.

Sintaks Regular Expression

Pada saat menggunakan regular expression, Anda akan mengenal karakter yang memiliki arti khusus, yaitu karakter escape (*escape character*). Karakter escape merupakan tanda backslash yang diikuti dengan karakter khusus atau kumpulan karakter. Untuk mengetahui karakter-karakter escape di dalam regular expression, perhatikan penjelasan berikut ini.

Karakter	Keterangan
\a	Karakter bell (alarm)
\b	Backspace
\t	Tabulator
\r	Return (carriage return)
\v	Tab vertikal
\f	Pindah halaman (<i>form feed</i>)
\n	Baris baru (<i>line feed</i>)
\e	Escape

Di samping karakter escape, Anda juga akan sering menggunakan perintah dalam bentuk kelas-kelas karakter. Di mana kelas karakter merupakan kumpulan karakter yang akan mencari kesesuaian atau kecocokan apabila ada salah satu dari karakter tercakup di dalam pola yang ditetapkan. Berikut ini adalah penjelasan mengenai perintah-perintah untuk mencocokkan karakter.

Perintah	Keterangan
[abcd]	Sesuai dengan semua karakter yang ada di dalam tanda kurung siku.
[^abc]	Sesuai dengan semua karakter yang tidak ada di dalam tanda kurung siku.
[0-9]	Tanda penghubung (minus) digunakan sebagai jangkauan karakter.
.	Tanda titik berarti sesuai dengan semua karakter, kecuali baris baru (\n).
\w	Sesuai dengan semua karakter atau kata, ekuivalen dengan [a-zA-Z_0-9].
\W	Sesuai dengan non-karakter, ekuivalen dengan [^a-zA-Z_0-9].
\s	Sesuai dengan karakter kosong, ekuivalen dengan \f, \n, \r, \t, dan \v.
\S	Tidak sesuai dengan semua karakter kosong.
\d	Sesuai dengan angka antara 0 sampai 9.
\D	Tidak sesuai dengan angka antara 0 sampai 9.

Selain karakter-karakter khusus di atas, ada dua buah metakarakter yang perlu Anda ketahui, yaitu karakter ^ dan \$. Karakter ^ berarti bahwa kesesuaian atau kecocokan harus dimulai dari awal string atau awal baris, sedangkan karakter \$ memiliki arti kecocokan pada akhir string.

Sebagai contoh, kita menetapkan bahwa tahun harus terdiri dari empat angka dan tidak boleh kurang dari tahun 2000. Kasus ini dapat kita selesaikan dengan menggunakan pola (20)\d\d. Bagaimana jika dua

angka juga diperkenankan? Kita perlu memperbaikinya, sehingga menjadi berikut:

```
(^(20)\d\d$)|(^d{2}$)
```

Ekspresi di atas terdiri dari dua subpola, pertama `(^(20)\d\d$)` dan subpola kedua adalah `(^d{2}$)`. Pada subpola pertama, kita memberikan karakter `^` yang berarti pencocokan dimulai dari awal, kemudian Anda harus menulis angka 20, dan dua angka sisanya sembarang. Terakhir kita tutup dengan karakter `$` yang berarti bahwa pencocokan juga akan dilakukan sampai akhir karakter, dalam hal ini adalah angka. Sebelum mendefinisikan subpola kedua, kita memberikan karakter `|` yang sama artinya dengan operator logika OR (atau). Ada pun subpola kedua juga kitaawali dengan karakter `^` dan diakhiri karakter `$`, sehingga dari awal sampai akhir angka harus Anda isi. Berikutnya karakter `\d` menandakan string berupa angka 0 sampai 9, dan angka 2 di dalam kurung kurawal memiliki arti bahwa persisnya adalah 2 angka.

Menggunakan Objek Regex

Dari beberapa kelas yang tersedia, kelas `Regex` dan `Match` merupakan kelas yang paling sering digunakan, terutama untuk membuat ekspresi sederhana. Kelas `Regex` adalah kelas yang merepresentasikan ekspresi *immutable* (*read-only*), dan digunakan untuk menciptakan objek `Regex`. Sementara itu, kelas `Match` merepresentasikan hasil regular expression yang sesuai dengan operasi.

Pada saat Anda menggunakan kelas-kelas regular expression, Anda perlu mengimport namespace di bagian atas kode program. Tujuannya adalah agar tidak perlu menuliskan nama lengkap kelas secara eksplisit, jadi cukup nama pendeknya (setelah nama namespace tersebut).

```
' Import namespace untuk kelas-kelas regex
Imports System.Text.RegularExpressions
```

Contoh penggunaan objek `Regex` dan `Match` diperlihatkan seperti berikut:

```
' Import namespace untuk kelas-kelas regex
Imports System.Text.RegularExpressions

Public Class Form1
    Inherits System.Windows.Forms.Form

    ' Windows Form Designer generated code

    Private Sub Button1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button1.Click
        ' Mendeklarasikan variabel objek dengan tipe Regex.
        Dim r As Regex
        ' Menciptakan objek Regex dan mendefinisikan regular expression
        r = New Regex("abc")
        ' Mencari string "abc" di dalam "123abc456"
        Dim m As Match = r.Match("123abc456")
        ' Mengembalikan True jika cocok
        If m.Success Then
            ' Mencetak posisi string yang ditemukan
            MsgBox("Ditemukan pada posisi " & m.Index.ToString())
        Else
            MsgBox("String Tidak Cocok")
        End If
        Console.ReadLine()
    End Sub

End Class
```

Pada kasus di atas, kita ingin mencocokkan apakah string **“abc”** berada di dalam string **“123abc456”**, dan menentukan posisinya jika ditemukan. Hasil yang kita dapatkan adalah string ditemukan, dan posisinya berada di index 3. Penghitungan nilai index dimulai dari 0, dan akan berhenti pada string pertama (huruf a). Sebaliknya, apabila pola **“abc”** diubah menjadi **“a bc”** atau **“a1bc”**, maka method `Match` akan mengembalikan nilai **False**, karena pola tidak sesuai dengan string target.

Ekspresi Validasi Data

Untuk melakukan validasi data, terlebih dahulu Anda harus membuat suatu ekspresi. Di mana ekspresi ini terbentuk dari susunan pola-pola sesuai dengan sintaks dasar penulisan regex. Melalui ekspresi inilah nantinya Anda dapat membandingkan, mengubah, mengganti, atau menghapus string. Regular expression memungkinkan Anda untuk melakukan validasi data melalui ekspresi-ekspresi yang telah Anda tentukan sebelumnya. Berdasarkan aturan yang ditetapkan, sebenarnya Anda juga dapat membuat pola yang lebih kompleks. Secara umum, ada beberapa ekspresi yang cukup sering membantu kita di dalam menyelesaikan

persoalan validasi.

- Ekspresi karakter numerik

Karakter numerik dituliskan dengan menggunakan ekspresi `[0-9]` atau `\d`. Ekspresi ini memiliki arti sesuai atau cocok dengan bilangan desimal tunggal antara 0 sampai 9.

- Ekspresi karakter non-numerik

Karakter non-numerik di sini diartikan sebagai semua karakter kecuali karakter angka. Cara penulisannya cukup singkat, yaitu menggunakan ekspresi `^[^0-9]` atau `\D`. Ingat kembali karakter `^` di dalam kurung siku pada pembahasan sebelumnya, di mana memiliki arti semua karakter kecuali angka 0 sampai 9. Perhatikan bahwa karakter `^` di dalam kurung siku bukan merupakan asersi seperti contoh kasus tahun.

- Ekspresi huruf kecil dan huruf besar

Huruf besar diekspresikan dengan karakter `[A-Z]`, sedangkan ekspresi huruf kecil adalah `[a-z]`. Ekspresi ini memiliki arti huruf antara a sampai dengan z, sehingga angka 0 sampai 9 tidak termasuk didalamnya. Jika ingin mendapatkan huruf besar dan huruf kecil, Anda tinggal menggabungkannya menjadi `[a-zA-Z]`.

- Ekspresi alamat IP

Sebagaimana Anda ketahui, alamat IP (*Internet Protocol*) versi 4 dituliskan seperti contoh berikut: 192.168.0.3. Ada pun untuk menyatakan alamat IP yang valid, kita menggunakan pola dalam ekspresi regex berikut:

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
```

Jika Anda perhatikan, ekspresi di atas mengandung tiga subpola yang dibatasi oleh tanda titik. Subpola pertama adalah ekspresi `\d{1,3}`, artinya angka 0 sampai 9 dan paling sedikit 1 angka serta tidak boleh lebih dari 3 angka. Tanda backslash digunakan untuk meloloskan tanda titik, sehingga didapatkan alamat IP yang valid. Ada pun tiga subpola lainnya sama dengan subpola pertama, hanya saja subpola terakhir tidak memerlukan tanda titik.

- Ekspresi alamat email

Alamat email secara umum dituliskan seperti contoh berikut: `didik_rpl@yahoo.com`. Dari sini Anda sudah dapat merancang pola yang tepat, yakni terdiri dari tiga subpola.

```
^([w-]+)@([w-]+\.)+[A-Za-z]{2,3}$
```

Untuk mendapatkan pola yang tepat, kita mengawali dengan karakter `^` dan menutup dengan karakter `$`. Berikutnya, subpola pertama menyatakan semua karakter diperbolehkan, dan dilanjutkan dengan karakter `@`. Setelah itu meloloskan tanda titik untuk digunakan sebagai akhir nama domain. Terakhir adalah penulisan TLD (*Top Level Domain*), di mana boleh dua tetapi tidak lebih dari 3 karakter.

Di dalam kelas `Regex` terdapat beberapa method static yang memungkinkan Anda untuk menggunakan regular expression tanpa secara eksplisit menciptakan objek `Regex`. Artinya, kita tidak harus menginstantiasi kelas `Regex` untuk menciptakan objek `Regex`, karena method static akan menciptakan objek secara implisit.

Berdasarkan ekspresi-ekspresi di atas, Anda dapat melakukan validasi melalui pola yang telah Anda definisikan. Untuk contoh kasus yang lebih lanjut, diharapkan masukan berupa tanggal dalam format `dd/mm/yyyy`. Di mana tanggal tidak boleh lebih dari 31 dan bulan juga tidak boleh lebih dari 12. Bagaimana cara menyelesaikannya? Perhatikan kode program berikut:

```
' Import namespace untuk kelas Regex
Imports System.Text.RegularExpressions

Public Class Form1
    Inherits System.Windows.Forms.Form

    Function IsValidDate(ByVal dt As String) As Boolean
        ' Tulis ekspresi dalam satu baris menyamping
        ' Mengembalikan True jika string sesuai dengan pola yang didefinisikan
        Return (Regex.IsMatch(dt, _
            "(^0[1-9]{1}|^[12][0-9]|3[01])-(0[1-9]|1[012])-(19|20)\d\d$"))
    End Function

    Private Sub Button1_Click(ByVal sender As System.Object, _
```

```
ByVal e As System.EventArgs) Handles Button1.Click
    Dim tgl1, tgl2 As String
    tgl1 = "30-06-2005"
    tgl2 = "2005-07-25"

    If IsValidDate(tgl1) Then
        Label1.Text = tgl1 + " Valid Date"
    Else
        Label1.Text = tgl1 + " Not Valid Date"
    End If

    If IsValidDate(tgl2) Then
        Label2.Text = tgl2 + " Valid Date"
    Else
        Label2.Text = tgl2 + " Not Valid Date"
    End If

End Sub

End Class
```

Pada contoh di atas, kita tidak menciptakan objek Regex secara eksplisit, tetapi menggunakan method `IsValidDate`, yang juga merupakan method statik. Hasil dari kode program di atas pasti sudah dapat Anda tebak, di mana string pertama merupakan penulisan format tanggal yang tepat, sedangkan string kedua tidak sesuai dengan pola Anda.

Validasi dengan Regular Expression

Setelah cukup memahami dasar-dasar regular expression, kita akan melakukan implementasi dalam rangka mendapatkan data yang sah. Untuk melengkapi validasi yang Anda buat, Anda juga dapat memanfaatkan kontrol `ErrorProvider`.

Sekarang siapkan project baru, kemudian masukkan empat buah kontrol `TextBox`, sebuah kontrol `ErrorProvider`, dan sebuah `Button`. Masing-masing `textbox` akan kita gunakan untuk masukan nama, tanggal, email, dan alamat URL. Kode program selengkapnya yang digunakan adalah sebagai berikut:

```
Imports System.Text.RegularExpressions

Public Class Form1
    Inherits System.Windows.Forms.Form

    ' Windows Form Designer generated code

    Private Sub Form1_Load(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        TextBox1.Focus()
        ' membatasi karakter tanggal
        TextBox2.MaxLength = 10
    End Sub

    Private Sub TextBox1_Leave(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles TextBox1.Leave
        If TextBox1.Text = "" Then
            TextBox1.Focus()
        End If
    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles TextBox1.TextChanged
        If Not IsAlpha(TextBox1.Text) Then
            ErrorProvider1.SetError(TextBox1, _
                "Isi dengan Huruf A-Z atau a-z" & Chr(10) & _
                "Satu kata saja ya.. ")
        Else
            ErrorProvider1.SetError(TextBox1, "")
        End If
    End Sub

    Private Sub TextBox2_TextChanged(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles TextBox2.TextChanged
        If Not IsValidDate(TextBox2.Text) Then
            ErrorProvider1.SetError(TextBox2, _
                "Isi dengan tgl-bln-th," & Chr(10) & _
                "misal: 01/12/2005")
        Else
            ErrorProvider1.SetError(TextBox2, "")
        End If
    End Sub
```

```
Private Sub TextBox3_TextChanged(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles TextBox3.TextChanged
    If Not IsValidEmail(TextBox3.Text) Then
        ErrorProvider1.SetError(TextBox3, _
            "Alamat Email Tidak Valid")
    Else
        ErrorProvider1.SetError(TextBox3, "")
    End If
End Sub

Private Sub TextBox4_TextChanged(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles TextBox4.TextChanged
    If Not IsValidURL(TextBox4.Text) Then
        ErrorProvider1.SetError(TextBox4, _
            "URL Tidak Valid")
    Else
        ErrorProvider1.SetError(TextBox4, "")
    End If
End Sub

' Fungsi untuk memeriksa input huruf (satu kata)
' mengembalikan nilai True jika string sesuai
Function IsAlpha(ByVal str As String) As Boolean
    Return (Regex.IsMatch(str, "[A-Za-z]+$"))
End Function

' Fungsi untuk memeriksa format tanggal
Function IsValidDate(ByVal tgl As String) As Boolean
    Return (Regex.IsMatch(tgl, _
        "(^0[1-9]{1}|^[12][0-9]|3[01])([/](0[1-9]|1[012])[/](19|20)\d\d$"))
End Function

' Fungsi untuk memeriksa format alamat email
Function IsValidEmail(ByVal email As String) As Boolean
    Return (Regex.IsMatch(email, "^[a-zA-Z0-9_+]{1,64}@[a-zA-Z0-9]{2,3}$"))
End Function

' Fungsi untuk memeriksa format URL
Function IsValidURL(ByVal url As String) As Boolean
    Return (Regex.IsMatch(url, "http://([a-zA-Z0-9]+[.])+[a-zA-Z0-9]+(/[a-zA-Z0-9_/?%&=]*)?"))
End Function

End Class
```

Pada validasi ini, Anda juga dapat menggunakan event-event TextBox lain yang mendukung, misalnya Leave, KeyPress, atau lainnya. Ada pun di sini kita menggunakan event TextChanged, karena memiliki waktu tanggap lebih cepat. Akibatnya ikon error juga akan lebih sering muncul, meskipun sebenarnya data yang Anda masukkan belum selesai dituliskan.

Gambar 1 Validasi menggunakan regular expression

Pada saat membuat ekspresi untuk suatu validasi, sangat mungkin tidak sama persis. Artinya ada beberapa cara yang dapat Anda pergunakan untuk membuat ekspresi, dengan catatan tetap memperhatikan sintaks dasar regular expression. Ketika Anda sudah memahami sintaks regular expression, dengan mudah Anda dapat membaca maksud dan tujuan suatu ekspresi. Dengan demikian, kiranya tidak sulit melakukan validasi data, jika kita mengetahui format yang kita kehendaki.

Semoga bermanfaat.