

Re: Regular expression optimization

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2006-01/msg01041.html>

- *From:* "Kevin Spencer" <kevin@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 26 Jan 2006 12:58:39 -0500
-

I have to correct myself. The code is fine, but my explanation has an error in it. I *initially* thought of using named groups, but as I continued to optimize the code, I realized that I didn't need to use named groups, as the position of the group in the GroupNames array would correspond to the position in the replacement array of strings, to identify the replacement string to use. You'll notice the MatchEvaluator delegate doesn't use the group names, but only iterates through the Collection.

--

HTH,

Kevin Spencer
Microsoft MVP
..Net Developer
Who is Mighty Abbott?
A twin turret scalawag.

"Kevin Spencer" <kevin@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:OiJoLhpIGHA.2896@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> Hi Billa,

>

> This was an interesting problem to me, so I took a crack at solving it.

>

> I constructed a class which uses the Regex.Replace overload taking an
> input string and a MatchEvaluator delegate. If you've never looked at this
> overload, the Regular Expression is evaluated against the input string,
> and the MatchEvaluator Delegate is called for each Match in the result.

>

> The first part required combining the separate Regular Expression strings
> into a single Regular Expression. I used grouping with the "|" (or)
> operator, so that the Regular Expression would match any of the Regular
> Expressions in the single expression.

>

> The trick was to get the MatchEvaluator delegate to recognize which of the
> sub-expressions formed the particular Match being passed to it. For this,
> I used named groups. The class has a private string array of replacement
> strings which is passed along with the Regular Expression to the
> Constructor. The replacement string array must match the number of

Re: Regular expression optimization

```
> sub-expressions in the Regular Expression, and in the same order.
>
> The Replace Method of the class loops through the array of Groups in the
> Match, which is the same as the array of Groups in the Regular Expression.
> It ignores the first group in the array, as that is always the match
> itself. It looks for the Group with the Success property as true, and
> returns the replacement string in the array that corresponds to that
> position in the Groups collection.
>
> I tested this, and it is bug-free. The class definition follows, followed
> by an example:
>
> /// <summary>
> /// Replaces multiple Regular Expressions in a string with multiple
> replacement
> /// strings without having to perform separate replacements
> /// </summary>
> /// <remarks>The <c>System.Text.RegularExpressions.Regex.Replace(string,
> MatchEvaluator)</c>
> /// overloads replace a single Regular Expression in a string. To replace
> many
> /// Regular Expressions in a string would entail creating many strings.
> This class
> /// enables the replacement to be done with a single string
> returned.</remarks>
> public class MultiReplacer
> {
>     private string[] Replacers;
>     private Regex r;
>
>     private string Replacer(Match m)
>     {
>         for(int i = 1; i < m.Groups.Count; i++)
>         {
>             if (m.Groups[i].Success)
>                 return Replacers[i - 1];
>         }
>         return "";
>     }
>
>     /// <summary>
>     /// Replaces all groups matching the expression initializer variable in
>     the input
>     /// string with the matching replacement string from the array of
>     replacement
>     /// strings passed in the initializer.
>     /// </summary>
>     /// <param name="input">String to evaluate.</param>
>     /// <returns>The fully-replaced string.</returns>
>     public string Replace(string input)
>     {
```

Re: Regular expression optimization

```
> MatchEvaluator meval = new MatchEvaluator(Replacer);
> return r.Replace(input, meval);
> }
>
> /// <summary type="System.String">
> /// Constructor.
> /// </summary>
> /// <param name="expression">Regular Expression String.</param>
> /// <param name="replacers">Array of replacement strings.</param>
> /// <remarks>The <paramref name="expression"/> parameter must be a
> Regular Expression using
> /// named groups, combined with "|" to match any of the groups in the
> /// <paramref name="expression"/>. The <paramref name="replacers"/>
> array
> /// must have the same number of elements as the number of named groups
> in the
> /// <paramref name="expression"/>, and in the same order.</remarks>
> public MultiReplacer(string expression,
> string[] replacers)
> {
> r = new Regex(expression);
> Replacers = replacers;
> }
> }
>
> example (from my test form):
>
> private void btnMultiReplace_Click(object sender, EventArgs e)
> {
> string s = @"(?<multistatus><a\.:multistatus
> [^>]*\>)|(?<endresponse>\</response\>)" ;
> string[] replacers = new string[] { @"<a:multistatus>",
> @"</nsResp:response>" };
> MultiReplacer replacer = new MultiReplacer(s, replacers);
>
> // Calls a method that adds the text to the same TextBox
> SetTextMessage(replacer.Replace(txtMessage.Text), false);
> }
>
> ---
> HTH,
>
> Kevin Spencer
> Microsoft MVP
> .Net Developer
> Who is Mighty Abbott?
> A twin turret scalawag.
>
>
> "Billa" <BillaTilla@xxxxxxxx> wrote in message
> <news:1138286413.033980.104010@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
```

Re: Regular expression optimization

>> Hi,
>> I am replacing a big string using different regular expressions (see
>> some example at the end of the message). The problem is whenever I
>> apply a "replace" it makes a new copy of string and I want to avoid
>> that. My question here is if there is a way to pass either a memory
>> stream or array of "find", "replace" expressions or any other way to
>> avoid multiple copies of a string.
>>
>> Any help will be highly appreciated
>>
>> Regards,
>> Billa
>>
>> Example:
>> 'Delete all namespaces from <multistatus> root element
>> Pattern = "\\<a\\:multistatus [^>]*\\>"
>> ReplaceWith = "<a:multistatus>"
>> sContents = Regex.Replace(sContents, Pattern, ReplaceWith,
>> RegexOptions.IgnorePatternWhitespace)
>>
>> 'Step#8-B.
>> 'doing the closing tag part
>> Pattern = "\\<\\response\\>"
>> ReplaceWith = "</nsResp:response>"
>> sContents = Regex.Replace(sContents, Pattern, ReplaceWith,
>> RegexOptions.IgnorePatternWhitespace)
>>
>
>

• *Follow-Ups:*

- ◆ [**Re: Regular expression optimization**](#)

◇ From: Billa

• *References:*

- ◆ [**Regular expression optimization**](#)

◇ From: Billa

- ◆ [**Re: Regular expression optimization**](#)

◇ From: Kevin Spencer

- Prev by Date: [**Re: converting IDL to .net Interfaces**](#)
- Next by Date: [**Re: Securing Software with License**](#)
- Previous by thread: [**Re: Regular expression optimization**](#)
- Next by thread: [**Re: Regular expression optimization**](#)
- Index(es):

- ◆ [**Date**](#)

Re: Regular expression optimization

◆ *Thread*