

## Re: ReplacerStream

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-10/msg00099.html>

---

- *From:* Peter Duniho <[NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx](mailto:NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 05 Oct 2007 01:21:03 -0700
- 

I'm sorry for the lack of replies. I myself didn't even notice your message until your third post, for some reason. Anyway, not sure I have any great answers, but here's a try...

hvj@xxxxxxxxxxx wrote:

To make my self more clear, for instance you should be able to replace all occurences of "wrongstring" with "rightstring". I think at least this would requiring buffering, because you can't tell whether you must replace untill all the bytes of the "wrongstring" have been read.

What I am doing now as a workaround is reading the stream into a string, do a replace on that string and create a stream again to be read base on that string. But this is not the most elegant and performing way:

```
string _strXml = null;
using (TextReader tr = new StreamReader(stream))
{
    _strXml = tr.ReadToEnd();
    _strXml = _strXml.Replace(wrongstring,rightstring);
    tr.Close();
}
```

Well, how long as these streams going to be? Can you read the entire stream immediately, or will there be a delay? If there's a delay, is it okay for the new stream to not be available until the original stream has been entirely read?

Whatever the answers to those questions, if the answers don't suggest a fundamental problem with the technique you're using, it may in fact be the best choice. The code is simple and easy to read, and I don't think performance should be too bad. The only real downsides I see are that you need the entire stream in memory first, and the output stream cannot even be started until the input stream is finished. If those aren't actually problems for your scenario, I'd say you're done. :)

If those are problems, and you are looking just for a single string, it seems to me that you could just read the stream one character at a time, checking to see if it matches the current character in your search string. You'd update the current character in the search string according to whether it matches; for a match, you'd increment by one, for a non-match you'd reset to the first character.

## Re: ReplacerStream

Anywhere in the stream when you have a potential match, you'd delay output of the data to the new stream until you've resolved the match, either by finding a character that doesn't match or by reaching the end of the search string having successfully matched all characters.

If it was an unsuccessful match, then you'd just emit the first character that hadn't already been sent to the output stream and start the search again at the next character (you need to do that, in case there are repeated characters or strings in your search string, so that if you hit, for example, "aaabb" when you're looking for "aabb" you don't wind up missing the matching pattern because you'd already consumed "aaa" by the time you figured out the match didn't start at the first "a"...if you have special knowledge of the input and search strings, you may be able to avoid having to reprocess characters).

If you have multiple search strings you want to process all at the same time, you might consider a state graph, built based on the search strings (i.e. using the characters in the strings to define the links between the graph nodes), and using the input characters to traverse the graph. You could still do multiple search strings using a variation of the previous method I describe, but if you understand a state graph, the implementation is more straightforward IMHO using one, than if you wrote the code to keep track of which string you're trying to match, especially if there's the potential for any overlap between strings (for example, if you could be looking for "abc" and "abd" at the same time).

Note that the logic to do the searching is somewhat independent of how to deal with the streams. However you do the search, you can either just build the search into whatever processing you already have for the input stream or you could, as it appears you're at least considering, create a new Stream-based class that filters the input data as it goes through.

How best to implement a custom Stream class depends on how exactly you want to use it. In fact, you might find that if you're specifically dealing with string data, it might make more sense to create a custom TextReader-based class instead. You could even derive from, for example, StreamReader, and just override the specific methods that handle reading from the stream. That way you don't have to bother implementing behaviors that are fine as is (like managing the buffering, positioning, etc.)

So, it seems you've really asked two questions: how to filter the data, and how to create a custom stream class. Hopefully there's some useful info above.

Pete

.