Subject: Re: Bug in the SAX parser?
Posted by Karl Schultz on Tue, 09 Jan 2007 16:40:46 GMT
View Forum Message <> Reply to Message

On Tue, 09 Jan 2007 06:34:58 -0800, mattf wrote:

> Just ran into a, uh, interesting problem with the SAX parser in IDL. At
> one point in processing an XML file, I have to remove the ignorable
> white space-- so I wrote a 'remove_white' subclass of the SAX parser to
> do that (a good exercise, btw, for anyone who's learning about this
> stuff). However, it turns out that the file I want to process has
> two-byte characters-- the 'remove_white' parser isn't bothered by that,
> but it appears to have a problem with recognizing a Windows newline
> (i.e., linefeed + carriage return) in two-byte character form.
> Specifically, it spots the carriage return, but not the linefeed-- so,
> my output file still has all the linefeeds. This is easy enough to deal
> with, in the subclassed 'Chars' method I test for
>
> chars[0] ne string(10B)
>
> before sending the string to the output file. But I think it's a bug.

If I look at a single-byte text file on Windows, the newline sequence is
carriage return + linefeed, not LF + CR, as you indicate.

This might be enough for the parser to behave as you describe.

Did the file undergo any newline translation as it may have been moved
to/from a Unix box? (It may have been treated as a binary file, and NOT
get the correct newline translation)

Does reversing the newline sequence help?

Subject: Re: Bug in the SAX parser?
Posted by Matt Feinstein on Tue, 09 Jan 2007 21:29:23 GMT
View Forum Message <> Reply to Message

In article <12q7hcegpr0gm6a@corp.supernews.com>, Karl Schultz
<k_remove_schultz@ittvis.com> wrote:

> On Tue, 09 Jan 2007 06:34:58 -0800, mattf wrote:
>
>> Just ran into a, uh, interesting problem with the SAX parser in IDL. At
>> one point in processing an XML file, I have to remove the ignorable
>> white space-- so I wrote a 'remove_white' subclass of the SAX parser to
>> do that (a good exercise, btw, for anyone who's learning about this
>> stuff). However, it turns out that the file I want to process has

>> two-byte characters-- the 'remove_white' parser isn't bothered by that,
>> but it appears to have a problem with recognizing a Windows newline
>> (i.e., linefeed + carriage return) in two-byte character form.
>> Specifically, it spots the carriage return, but not the linefeed-- so,
>> my output file still has all the linefeeds. This is easy enough to deal
>> with, in the subclassed 'Chars' method I test for
>>
>> chars[0] ne string(10B)
>>
>> before sending the string to the output file. But I think it's a bug.
>
> If I look at a single-byte text file on Windows, the newline sequence is
> carriage return + linefeed, not LF + CR, as you indicate.
>
> This might be enough for the parser to behave as you describe.
>
> Did the file undergo any newline translation as it may have been moved
> to/from a Unix box? (It may have been treated as a binary file, and NOT
> get the correct newline translation)
>
> Does reversing the newline sequence help?

Interesting suggestion. I'll look at that tomorrow.

Subject: Re: Bug in the SAX parser?
Posted by Matt Feinstein on Wed, 10 Jan 2007 21:49:26 GMT
View Forum Message <> Reply to Message

In article <09012007162923276%mfein@clark.net>, Matt Feinstein
<mfein@clark.net> wrote:

> In article <12q7hcegpr0gm6a@corp.supernews.com>, Karl Schultz
> <k_remove_schultz@ittvis.com> wrote:
>
>> On Tue, 09 Jan 2007 06:34:58 -0800, mattf wrote:
>>
>>> Just ran into a, uh, interesting problem with the SAX parser in IDL. At
>>> one point in processing an XML file, I have to remove the ignorable
>>> white space-- so I wrote a 'remove_white' subclass of the SAX parser to
>>> do that (a good exercise, btw, for anyone who's learning about this
>>> stuff). However, it turns out that the file I want to process has
>>> two-byte characters-- the 'remove_white' parser isn't bothered by that,
>>> but it appears to have a problem with recognizing a Windows newline
>>> (i.e., linefeed + carriage return) in two-byte character form.
>>> Specifically, it spots the carriage return, but not the linefeed-- so,
>>> my output file still has all the linefeeds. This is easy enough to deal
>>> with, in the subclassed 'Chars' method I test for

>>>
>>> chars[0] ne string(10B)
>>>
>>> before sending the string to the output file. But I think it's a bug.
>>
>> If I look at a single-byte text file on Windows, the newline sequence is
>> carriage return + linefeed, not LF + CR, as you indicate.
>>
>> This might be enough for the parser to behave as you describe.
>>
>> Did the file undergo any newline translation as it may have been moved
>> to/from a Unix box? (It may have been treated as a binary file, and NOT
>> get the correct newline translation)
>>
>> Does reversing the newline sequence help?
>
> Interesting suggestion. I'll look at that tomorrow

I checked the original file, and it looks like the carriage return and
line feed are in the correct order, so the binary form of the four-byte
new line is:

00 0D 00 0A
