

## Re: XMLReader skip current element

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg00838.html>

---

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx>
  - *Date:* Tue, 05 Jun 2007 14:47:01 -0700
- 

On Tue, 05 Jun 2007 13:40:01 -0700, Jon Skeet [C# MVP] <skeet@xxxxxxxx> wrote:

[...]

You should try Eclipse some time – it will compile (in some cases, at least) syntactically invalid code, generating code which throws an exception when it's got to somewhere that the compilation broke. Not terribly handy, but quite cute.

Well, sure. I can appreciate "cute". :) But as you say, not terribly handy. Likewise, just how handy would it be to just skip over an invalid section of XML, when you have no idea what the overall effect of doing so would be? Just because the remaining XML can be parsed, that doesn't mean that it can be \*used\* without the part that was erroneous.

[...] Certainly the VS

2005 XML editor was able to automatically close the "blech" tag in the below XML, despite the previous error:

I certainly agree that it \*can\* be done. I just am not convinced it makes sense to bother writing the code to do so. It does seem to me that in an editor, where the user is actively modifying the data, it makes more sense to put the effort in, but even there I wouldn't necessarily insist on it (even in VS there are limits to what it can recover from, and frankly it only handles the simplest situations). I expect it's something you see in editors that are intended to be feature-laden, considered "heavy-duty" (that's certainly how I'd describe VS).

In a situation where the data is static though, I don't see the use in recovering. You never know when the data that was in error was critical to the use of the larger XML document. Just because you can successfully parse the rest of the document doesn't mean you should, just as just because a compiler could make an assumption about where to insert a missing semi-colon doesn't mean it should.

Pete

.