Parsing Broken XM

The XML specification mandates that all conforming XML parsers employ "draconian error handling." That is, they must halt and catch fire as soon as they detect any sort of wellformedness error in the XML document. Wellformedness errors include mismatched start and end tags, undefined entities, illegal Unicode characters, and a number of other esoteric rules. This is in stark contrast to other common formats like HTML — your browser doesn't stop rendering a web page if you forget to close an HTML tag or escape an ampersand in an attribute value. (It is a common misconception that html has no defined error handling. HTML error handling is actually quite well-defined, but it's significantly more complicated than "halt and catch fire on first error.")

Some people (myself included) believe that it was a mistake for the inventors of XML to mandate draconian error handling. Don't get me wrong; I can certainly see the allure of simplifying the error handling rules. But in practice, the concept of "wellformedness" is trickier than it sounds, especially for XML documents (like Atom feeds) that are published on the web and served over HTTP. Despite the maturity of XML, which standardized on draconian error handling in 1997, surveys continually show a significant fraction of Atom feeds on the web are plagued with wellformedness errors.

So, I have both theoretical and practical reasons to parse XML documents "at any cost," that is, *not* to halt and catch fire at the first wellformedness error. If you find yourself wanting to do this too, `lxml` can help.

Here is a fragment of a broken XML document. I've highlighted the wellformedness error.

```
<?xml version='1.0' encoding='utf-8'?>
<feed xmlns='http://www.w3.org/2005/Atom' xml:lang='en'>
  <title>dive into &hellip;</title> <!-- error: "&hellip;" -->
...
```

```
</feed>
```

That’s an error, because the `&hellip`; entity is not defined in XML. (It is defined in HTML.) If you try to parse this broken feed with the default settings, `lxml` will choke on the undefined entity.

```
import lxml.etree
tree = lxml.etree.parse('feed-broken.xml')
#Traceback (most recent call last):
#  File "<stdin>", line 1, in <module>
#  File "lxml.etree.pyx", line 2693, in lxml.etree.parse (src/lxml/lxml.etree.c:52591)
#  File "parser.pxi", line 1478, in lxml.etree._parseDocument (src/lxml/lxml.etree.c:75665)
#  File "parser.pxi", line 1507, in lxml.etree._parseDocumentFromURL (src/lxml/lxml.etree.c:7
#  File "parser.pxi", line 1407, in lxml.etree._parseDocFromFile (src/lxml/lxml.etree.c:75002
#  File "parser.pxi", line 965, in lxml.etree._BaseParser._parseDocFromFile (src/lxml/lxml.et
#  File "parser.pxi", line 539, in lxml.etree._ParserContext._handleParseResultDoc (src/lxml/
#  File "parser.pxi", line 625, in lxml.etree._handleParseResult (src/lxml/lxml.etree.c:68877
#  File "parser.pxi", line 565, in lxml.etree._raiseParseError (src/lxml/lxml.etree.c:68125)
#lxml.etree.XMLSyntaxError: Entity 'hellip' not defined, line 3, column 28
```

To parse this broken XML document, despite its wellformedness error, you need to create a custom XML parser.

```
parser = lxml.etree.XMLParser(recover=True)                         #①
tree = lxml.etree.parse('feed-broken.xml', parser)                  #②
print (parser.error_log )                                           #③
#examples/feed-broken.xml:3:28:FATAL:PARSER:ERR_UNDECLARED_ENTITY: Entity 'hellip' not define

print (tree.findall('{http://www.w3.org/2005/Atom}title'))
#[<Element {http://www.w3.org/2005/Atom}title at ead510>]

title = tree.findall('{http://www.w3.org/2005/Atom}title')[0]
print (title.text )                                                 #④
#dive into

print(lxml.etree.tounicode(tree.getroot()))                         #⑤
#<feed xmlns='http://www.w3.org/2005/Atom' xml:lang='en'>
#  <title>dive into </title>
#.
# [rest of serialization snipped for brevity]
#.
```

① To create a custom parser, instantiate the `lxml.etree.XMLParser` class. It can take a number of different named arguments. The one we’re interested in here is the `recover` argument. When set to `True`, the XML parser will try its best to “recover” from wellformedness errors.

② To parse an XML document with your custom parser, `pass` the parser object as the second argument to the `parse()` function. Note that `lxml` does not raise an exception about the undefined `&hellip;` entity.

③ The parser keeps a log of the wellformedness errors that it has encountered. (This is actually true regardless of whether it is set to recover from those errors or not.)

④ Since it didn't know what to do with the undefined `&hellip;` entity, the parser just silently dropped it. The text content of the `title` element becomes `'dive into '`.

⑤ As you can see from the serialization, the `&hellip;` entity didn't get moved; it was simply dropped.

It is important to reiterate that there is **no guarantee of interoperability** with "recovering" XML parsers. A different parser might decide that it recognized the `&hellip;` entity from HTML, and replace it with `&amp;hellip;` instead. Is that "better"? Maybe. Is it "more correct"? No, they are both equally incorrect. The correct behavior (according to the XML specification) is to halt and catch fire. If you've decided not to do that, you're on your own.