20.8. `xml.dom.pulldom` — Support for building partial DOM trees

Source code: Lib/xml/dom/pulldom.py

The `xml.dom.pulldom` module provides a "pull parser" which can also be asked to produce DOM-accessible fragments of the document where necessary. The basic concept involves pulling "events" from a stream of incoming XML and processing them. In contrast to SAX which also employs an event-driven processing model together with callbacks, the user of a pull parser is responsible for explicitly pulling events from the stream, looping over those events until either processing is finished or an error condition occurs.

Warning: The `xml.dom.pulldom` module is not secure against maliciously constructed data. If you need to parse untrusted or unauthenticated data see XML vulnerabilities.

Example:

```
from xml.dom import pulldom

doc = pulldom.parse('sales_items.xml')
for event, node in doc:
    if event == pulldom.START_ELEMENT and node.tagName == 'item':
        if int(node.getAttribute('price')) > 50:
            doc.expandNode(node)
            print(node.toxml())
```

`event` is a constant and can be one of:

- `START_ELEMENT`
- `END_ELEMENT`
- `COMMENT`
- `START_DOCUMENT`
- `END_DOCUMENT`
- `CHARACTERS`
- `PROCESSING_INSTRUCTION`
- `IGNORABLE_WHITESPACE`

`node` is an object of type `xml.dom.minidom.Document`, `xml.dom.minidom.Element` or `xml.dom.minidom.Text`.

Since the document is treated as a “flat” stream of events, the document “tree” is implicitly traversed and the desired elements are found regardless of their depth in the tree. In other words, one does not need to consider hierarchical issues such as recursive searching of the document nodes, although if the context of elements were important, one would either need to maintain some context-related state (i.e. remembering where one is in the document at any given point) or to make use of the `DOMEventStream.expandNode()` method and switch to DOM-related processing.

class `xml.dom.pulldom.`**`PullDom`**(*documentFactory=None*)

Subclass of `xml.sax.handler.ContentHandler`.

class `xml.dom.pulldom.`**`SAX2DOM`**(*documentFactory=None*)

Subclass of `xml.sax.handler.ContentHandler`.

`xml.dom.pulldom.`**`parse`**(*stream_or_string*, *parser=None*, *bufsize=None*)

Return a `DOMEventStream` from the given input. *stream_or_string* may be either a file name, or a file-like object. *parser*, if given, must be an `XMLReader` object. This function will change the document handler of the parser and activate namespace support; other parser configuration (like setting an entity resolver) must have been done in advance.

If you have XML in a string, you can use the `parseString()` function instead:

`xml.dom.pulldom.`**`parseString`**(*string*, *parser=None*)

Return a `DOMEventStream` that represents the (Unicode) *string*.

`xml.dom.pulldom.`**`default_bufsize`**

Default value for the *bufsize* parameter to `parse()`.

The value of this variable can be changed before calling `parse()` and the new value will take effect.

20.8.1. DOMEventStream Objects

class `xml.dom.pulldom.`**`DOMEventStream`**(*stream*, *parser*, *bufsize*)

`getEvent`()

Return a tuple containing *event* and the current *node* as `xml.dom.minidom.Document` if event equals `START_DOCUMENT`, `xml.dom.minidom.Element` if event equals `START_ELEMENT` or `END_ELEMENT` or `xml.dom.minidom.Text` if event equals `CHARACTERS`. The current node does not contain information about its children, unless `expandNode()` is called.

expandNode(*node*)

Expands all children of *node* into *node*. Example:

```
from xml.dom import pulldom

xml = '<html><title>Foo</title> <p>Some text <div>and more</div
doc = pulldom.parseString(xml)
for event, node in doc:
    if event == pulldom.START_ELEMENT and node.tagName == 'p':
        # Following statement only prints '<p/>'
        print(node.toxml())
        doc.expandNode(node)
        # Following statement prints node with all its children
        print(node.toxml())
```

reset()