20.11. `xml.sax.saxutils` — SAX Utilities

Source code: Lib/xml/sax/saxutils.py

The module `xml.sax.saxutils` contains a number of classes and functions that are commonly useful when creating SAX applications, either in direct use, or as base classes.

`xml.sax.saxutils.`**`escape`**(*data*, *entities={}*)

Escape `'&'`, `'<'`, and `'>'` in a string of data.

You can escape other strings of data by passing a dictionary as the optional *entities* parameter. The keys and values must all be strings; each key will be replaced with its corresponding value. The characters `'&'`, `'<'` and `'>'` are always escaped, even if *entities* is provided.

`xml.sax.saxutils.`**`unescape`**(*data*, *entities={}*)

Unescape `'&'`, `'<'`, and `'>'` in a string of data.

You can unescape other strings of data by passing a dictionary as the optional *entities* parameter. The keys and values must all be strings; each key will be replaced with its corresponding value. `'&amp'`, `'<'`, and `'>'` are always unescaped, even if *entities* is provided.

`xml.sax.saxutils.`**`quoteattr`**(*data*, *entities={}*)

Similar to `escape()`, but also prepares *data* to be used as an attribute value. The return value is a quoted version of *data* with any additional required replacements. `quoteattr()` will select a quote character based on the content of *data*, attempting to avoid encoding any quote characters in the string. If both single- and double-quote characters are already in *data*, the double-quote characters will be encoded and *data* will be wrapped in double-quotes. The resulting string can be used directly as an attribute value:

```
>>> print("<element attr=%s>" % quoteattr("ab ' cd \" ef"))
<element attr="ab ' cd &quot; ef">
```

This function is useful when generating attribute values for HTML or any SGML using the reference concrete syntax.

class `xml.sax.saxutils.`**`XMLGenerator`**(*out=None*, *encoding='iso-8859-1'*, *short_empty_elements=False*)

This class implements the `ContentHandler` interface by writing SAX events back into an XML document. In other words, using an `XMLGenerator` as the content handler will reproduce the original document being parsed. *out* should be a file-like object which will default to *sys.stdout*. *encoding* is the encoding of the output stream which defaults to `'iso-8859-1'`. *short_empty_elements* controls the formatting of elements that contain no content: if `False` (the default) they are emitted as a pair of start/end tags, if set to `True` they are emitted as a single self-closed tag.

New in version 3.2: The *short_empty_elements* parameter.

class `xml.sax.saxutils.`**`XMLFilterBase`**(*base*)

This class is designed to sit between an `XMLReader` and the client application's event handlers. By default, it does nothing but pass requests up to the reader and events on to the handlers unmodified, but subclasses can override specific methods to modify the event stream or the configuration requests as they pass through.

`xml.sax.saxutils.`**`prepare_input_source`**(*source*, *base=''*)

This function takes an input source and an optional base URL and returns a fully resolved `InputSource` object ready for reading. The input source can be given as a string, a file-like object, or an `InputSource` object; parsers will use this function to implement the polymorphic *source* argument to their `parse()` method.