20.12. `xml.sax.xmlreader` — Interface for XML parsers

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

SAX parsers implement the `XMLReader` interface. They are implemented in a Python module, which must provide a function `create_parser()`. This function is invoked by `xml.sax.make_parser()` with no arguments to create a new parser object.

class `xml.sax.xmlreader.`**`XMLReader`**

Base class which can be inherited by SAX parsers.

class `xml.sax.xmlreader.`**`IncrementalParser`**

In some cases, it is desirable not to parse an input source at once, but to feed chunks of the document as they get available. Note that the reader will normally not read the entire file, but read it in chunks as well; still `parse()` won't return until the entire document is processed. So these interfaces should be used if the blocking behaviour of `parse()` is not desirable.

When the parser is instantiated it is ready to begin accepting data from the feed method immediately. After parsing has been finished with a call to close the reset method must be called to make the parser ready to accept new data, either from feed or using the parse method.

Note that these methods must *not* be called during parsing, that is, after parse has been called and before it returns.

By default, the class also implements the parse method of the XMLReader interface using the feed, close and reset methods of the IncrementalParser interface as a convenience to SAX 2.0 driver writers.

class `xml.sax.xmlreader.`**`Locator`**

Interface for associating a SAX event with a document location. A locator object will return valid results only during calls to DocumentHandler methods; at any other time, the results are unpredictable. If information is not available, methods may return `None`.

class `xml.sax.xmlreader.`**`InputSource`**(*system_id=None*)

Encapsulation of the information needed by the `XMLReader` to read entities.

This class may include information about the public identifier, system identifier, byte stream (possibly with character encoding information) and/or the character stream of an entity.

Applications will create objects of this class for use in the `XMLReader.parse()` method and for returning from EntityResolver.resolveEntity.

An `InputSource` belongs to the application, the `XMLReader` is not allowed to modify `InputSource` objects passed to it from the application, although it may make copies and modify those.

class `xml.sax.xmlreader.`**`AttributesImpl`**(*attrs*)

This is an implementation of the `Attributes` interface (see section The Attributes Interface). This is a dictionary-like object which represents the element attributes in a `startElement()` call. In addition to the most useful dictionary operations, it supports a number of other methods as described by the interface. Objects of this class should be instantiated by readers; *attrs* must be a dictionary-like object containing a mapping from attribute names to attribute values.

class `xml.sax.xmlreader.`**`AttributesNSImpl`**(*attrs*, *qnames*)

Namespace-aware variant of `AttributesImpl`, which will be passed to `startElementNS()`. It is derived from `AttributesImpl`, but understands attribute names as two-tuples of *namespaceURI* and *localname*. In addition, it provides a number of methods expecting qualified names as they appear in the original document. This class implements the `AttributesNS` interface (see section The AttributesNS Interface).

20.12.1. XMLReader Objects

The `XMLReader` interface supports the following methods:

`XMLReader.`**`parse`**(*source*)

Process an input source, producing SAX events. The *source* object can be a system identifier (a string identifying the input source – typically a file name or a URL), a file-like object, or an `InputSource` object. When `parse()` returns, the input is completely processed, and the parser object can be discarded or reset.

Changed in version 3.5: Added support of character streams.

`XMLReader.`**`getContentHandler`**()

Return the current `ContentHandler`.

`XMLReader.`**`setContentHandler`**(*handler*)

Set the current `ContentHandler`. If no `ContentHandler` is set, content events will be discarded.

`XMLReader.`**`getDTDHandler`**()

Return the current `DTDHandler`.

`XMLReader.`**`setDTDHandler`**(*handler*)

Set the current `DTDHandler`. If no `DTDHandler` is set, DTD events will be discarded.

`XMLReader.`**`getEntityResolver`**()

Return the current `EntityResolver`.

`XMLReader.`**`setEntityResolver`**(*handler*)

Set the current `EntityResolver`. If no `EntityResolver` is set, attempts to resolve an external entity will result in opening the system identifier for the entity, and fail if it is not available.

`XMLReader.`**`getErrorHandler`**()

Return the current `ErrorHandler`.

`XMLReader.`**`setErrorHandler`**(*handler*)

Set the current error handler. If no `ErrorHandler` is set, errors will be raised as exceptions, and warnings will be printed.

`XMLReader.`**`setLocale`**(*locale*)

Allow an application to set the locale for errors and warnings.

SAX parsers are not required to provide localization for errors and warnings; if they cannot support the requested locale, however, they must raise a SAX exception. Applications may request a locale change in the middle of a parse.

`XMLReader.`**`getFeature`**(*featurename*)

Return the current setting for feature *featurename*. If the feature is not recognized, `SAXNotRecognizedException` is raised. The well-known featurenames are listed in the module `xml.sax.handler`.

`XMLReader.`**`setFeature`**(*featurename*, *value*)

Set the *featurename* to *value*. If the feature is not recognized, `SAXNotRecognizedException` is raised. If the feature or its setting is not supported by the parser, *SAXNotSupportedException* is raised.

`XMLReader.`**`getProperty`**(*propertyname*)

Return the current setting for property *propertyname*. If the property is not recognized, a `SAXNotRecognizedException` is raised. The well-known propertynames are listed in the module `xml.sax.handler`.

`XMLReader.`**`setProperty`**(*propertyname*, *value*)

Set the *propertyname* to *value*. If the property is not recognized, `SAXNotRecognizedException` is raised. If the property or its setting is not supported by the parser, *SAXNotSupportedException* is raised.

20.12.2. IncrementalParser Objects

Instances of `IncrementalParser` offer the following additional methods:

`IncrementalParser.`**`feed`**(*data*)

Process a chunk of *data*.

`IncrementalParser.`**`close`**()

Assume the end of the document. That will check well-formedness conditions that can be checked only at the end, invoke handlers, and may clean up resources allocated during parsing.

`IncrementalParser.`**`reset`**()

This method is called after close has been called to reset the parser so that it is ready to parse new documents. The results of calling parse or feed after close without calling reset are undefined.

20.12.3. Locator Objects

Instances of `Locator` provide these methods:

`Locator.`**`getColumnNumber`**()

Return the column number where the current event begins.

`Locator.`**`getLineNumber`**()

Return the line number where the current event begins.

`Locator.`**`getPublicId`**()

Return the public identifier for the current event.

`Locator.`**`getSystemId`**()

Return the system identifier for the current event.

20.12.4. InputSource Objects

`InputSource.`**`setPublicId`**(*id*)

Sets the public identifier of this `InputSource`.

`InputSource.`**`getPublicId`**()

Returns the public identifier of this `InputSource`.

`InputSource.`**`setSystemId`**(*id*)

Sets the system identifier of this `InputSource`.

`InputSource.`**`getSystemId`**()

Returns the system identifier of this `InputSource`.

`InputSource.`**`setEncoding`**(*encoding*)

Sets the character encoding of this `InputSource`.

The encoding must be a string acceptable for an XML encoding declaration (see section 4.3.3 of the XML recommendation).

The encoding attribute of the `InputSource` is ignored if the `InputSource` also contains a character stream.

`InputSource.`**`getEncoding`**()

Get the character encoding of this InputSource.

`InputSource.`**`setByteStream`**(*bytefile*)

Set the byte stream (a binary file) for this input source.

The SAX parser will ignore this if there is also a character stream specified, but it will use a byte stream in preference to opening a URI connection itself.

If the application knows the character encoding of the byte stream, it should set it with the setEncoding method.

`InputSource.`**`getByteStream`**()

Get the byte stream for this input source.

The getEncoding method will return the character encoding for this byte stream, or `None` if unknown.

`InputSource.`**`setCharacterStream`**(*charfile*)

Set the character stream (a text file) for this input source.

If there is a character stream specified, the SAX parser will ignore any byte stream and will not attempt to open a URI connection to the system identifier.

`InputSource.`**`getCharacterStream`**()

Get the character stream for this input source.

20.12.5. The `Attributes` Interface

`Attributes` objects implement a portion of the mapping protocol, including the methods `copy()`, `get()`, `__contains__()`, `items()`, `keys()`, and `values()`. The following methods are also provided:

`Attributes.`**`getLength`**()

Return the number of attributes.

`Attributes.`**`getNames`**()

Return the names of the attributes.

`Attributes.`**`getType`**(*name*)

Returns the type of the attribute *name*, which is normally `'CDATA'`.

`Attributes.`**`getValue`**(*name*)

Return the value of attribute *name*.

20.12.6. The `AttributesNS` Interface

This interface is a subtype of the `Attributes` interface (see section The Attributes Interface). All methods supported by that interface are also available on `AttributesNS` objects.

The following methods are also available:

`AttributesNS.`**`getValueByQName`**(*name*)

Return the value for a qualified name.

`AttributesNS.`**`getNameByQName`**(*name*)

Return the `(namespace, localname)` pair for a qualified *name*.

`AttributesNS.`**`getQNameByName`**(*name*)

Return the qualified name for a `(namespace, localname)` pair.

`AttributesNS.`**`getQNames`**()

Return the qualified names of all attributes.