11 - Dec - 08

it has 11 methods

3 - get Type of a attribute
3 - get Value of a attribute

2 - get Name of attribute
2 - get index of attribute
1 - get length
1 - get URI

org.xml.sax

Interface Attributes

All Known Subinterfaces:

Attributes2

All Known Implementing Classes:

Attributes

5 - methods has index as argument (getting .. type, localName, qName, value, URI)
3 - methods has qName as arument (Getting ... type, value, index)
3 - methods has uri, localName as argument (Getting .... type, value, index)

`public interface` **`Attributes`**

the easiest way use INDEX based methods

Interface for a list of XML attributes.

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* **NO WARRANTY**. See http://www.saxproject.org for further information.~~

This interface allows access to a list of attributes in three different ways:

1. by attribute index;
2. by Namespace-qualified name; or
3. by qualified (prefixed) name.

localNAme, Uri... TWO argument

qName ..single argument

~~The list will not contain attributes that were declared #IMPLIED but not specified in the start tag. It will also not contain attributes used as Namespace declarations (xmlns*) unless the `http://xml.org/sax/features/namespace-prefixes` feature is set to *true* (it is *false* by default). Because SAX2 conforms to the original "Namespaces in XML" recommendation, it normally does not give namespace declaration attributes a namespace URI.~~

~~Some SAX2 parsers may support using an optional feature flag (`http://xml.org/sax/features/xmlns-uris`) to request that those attributes be given URIs, conforming to a later backwards-incompatible revision of that recommendation. (The attribute's "local name" will be the prefix, or "xmlns" when defining a default element namespace.) For portability, handler code should~~

~~always resolve that conflict, rather than requiring parsers that can change the setting of that feature flag.~~

~~If the namespace-prefixes feature (see above) is *false*, access by qualified name may not be available; if the `http://xml.org/sax/features/namespaces` feature is *false*, access by Namespace-qualified names may not be available.~~

~~This interface replaces the now-deprecated SAX1 `AttributeList` interface, which does not contain Namespace support. In addition to Namespace support, it adds the *getIndex* methods (below).~~

~~The order of attributes in the list is unspecified, and will vary from implementation to implementation.~~

~~**Since:**~~

~~SAX 2.0~~

~~**Version:**~~

~~2.0.1 (sax2r2)~~

~~**Author:**~~

~~David Megginson~~

~~**See Also:**~~

~~`AttributesImpl`, `DeclHandler.attributeDecl(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String)`~~

Method Summary

`int`	**`getIndex`**`(java.lang.String qName)` Look up the index of an attribute by XML qualified (prefixed) name.
`int`	**`getIndex`**`(java.lang.String uri, java.lang.String localName)` Look up the index of an attribute by Namespace name.
`int`	**`getLength`**`()` Return the number of attributes in the list.
`java.lang.String`	**`getLocalName`**`(int index)` Look up an attribute's local name by index.
`java.lang.String`	**`getQName`**`(int index)` Look up an attribute's XML qualified (prefixed) name by index.

java.lang.String	**getType**(int index) Look up an attribute's type by index.
java.lang.String	**getType**(java.lang.String qName) Look up an attribute's type by XML qualified (prefixed) name.
java.lang.String	**getType**(java.lang.String uri, java.lang.String localName) Look up an attribute's type by Namespace name.
java.lang.String	**getURI**(int index) Look up an attribute's Namespace URI by index.
java.lang.String	**getValue**(int index) Look up an attribute's value by index.
java.lang.String	**getValue**(java.lang.String qName) Look up an attribute's value by XML qualified (prefixed) name.
java.lang.String	**getValue**(java.lang.String uri, java.lang.String localName) Look up an attribute's value by Namespace name.

Method Detail

getLength

`int getLength()`

Return the number of attributes in the list.

Once you know the number of attributes, you can iterate through the list.

Returns:
The number of attributes in the list.

See Also:
getURI(int), getLocalName(int), getQName(int), getType(int), getValue(int)

---

getURI

`java.lang.String` **`getURI`**`(int index)`

Look up an attribute's Namespace URI by index.

Parameters:
`index` - The attribute index (zero-based).

Returns:
The Namespace URI, or the empty string if none is available, or null if the index is out of range.

See Also:
`getLength()`

---

getLocalName

`java.lang.String` **`getLocalName`**`(int index)`

Look up an attribute's local name by index.

Parameters:
`index` - The attribute index (zero-based).

Returns:
The local name, or the empty string if Namespace processing is not being performed, or null if the index is out of range.

See Also:
`getLength()`

---

getQName

`java.lang.String` **`getQName`**`(int index)`

Look up an attribute's XML qualified (prefixed) name by index.

Parameters:

`index` - The attribute index (zero-based).

Returns:

The XML qualified name, or the empty string if none is available, or null if the index is out of range.

See Also:

`getLength()`

---

getType

`java.lang.String` **`getType`**`(int index)`

Look up an attribute's type by index.

The attribute type is one of the strings "CDATA", "ID", "IDREF", "IDREFS", "NMTOKEN", "NMTOKENS", "ENTITY", "ENTITIES", or "NOTATION" (always in upper case).

If the parser has not read a declaration for the attribute, or if the parser does not report attribute types, then it must return the value "CDATA" as stated in the XML 1.0 Recommendation (clause 3.3.3, "Attribute-Value Normalization").

For an enumerated attribute that is not a notation, the parser will report the type as "NMTOKEN".

Parameters:

`index` - The attribute index (zero-based).

Returns:

The attribute's type as a string, or null if the index is out of range.

See Also:

`getLength()`

---

getValue

`java.lang.String` **`getValue`**`(int index)`

Look up an attribute's value by index.

If the attribute value is a list of tokens (IDREFS, ENTITIES, or NMTOKENS), the tokens will be concatenated into a single string with each token separated by a single space.

Parameters:
`index` - The attribute index (zero-based).

Returns:
The attribute's value as a string, or null if the index is out of range.

See Also:
`getLength()`

---

getIndex

```
int getIndex(java.lang.String uri,
             java.lang.String localName)
```

Look up the index of an attribute by Namespace name.

Parameters:
`uri` - The Namespace URI, or the empty string if the name has no Namespace URI.
`localName` - The attribute's local name.

Returns:
The index of the attribute, or -1 if it does not appear in the list.

---

getIndex

```
int getIndex(java.lang.String qName)
```

Look up the index of an attribute by XML qualified (prefixed) name.

Parameters:
`qName` - The qualified (prefixed) name.

Returns:
The index of the attribute, or -1 if it does not appear in the list.

---

getType

```
java.lang.String getType(java.lang.String uri,
                         java.lang.String localName)
```

Look up an attribute's type by Namespace name.

See `getType(int)` for a description of the possible types.

Parameters:

`uri` - The Namespace URI, or the empty String if the name has no Namespace URI.

`localName` - The local name of the attribute.

Returns:

The attribute type as a string, or null if the attribute is not in the list or if Namespace processing is not being performed.

---

getType

```
java.lang.String getType(java.lang.String qName)
```

Look up an attribute's type by XML qualified (prefixed) name.

See `getType(int)` for a description of the possible types.

Parameters:

`qName` - The XML qualified name.

Returns:

The attribute type as a string, or null if the attribute is not in the list or if qualified names are not available.

---

getValue

```
java.lang.String getValue(java.lang.String uri,
                          java.lang.String localName)
```

Look up an attribute's value by Namespace name.

See `getValue(int)` for a description of the possible values.

Parameters:

`uri` - The Namespace URI, or the empty String if the name has no Namespace URI.

`localName` - The local name of the attribute.

Returns:

The attribute value as a string, or null if the attribute is not in the list.

---

getValue

`java.lang.String` **`getValue`**`(java.lang.String qName)`

Look up an attribute's value by XML qualified (prefixed) name.

See `getValue(int)` for a description of the possible values.

Parameters:

`qName` - The XML qualified name.

Returns:

The attribute value as a string, or null if the attribute is not in the list or if qualified names are not available.

---

http://www-labs.det.uvigo.es/documentation/LRO/jaxp/jaxp-1_3-html/index.html

for examples for DOM
http://www.java-tips.org/org.w3c.dom/

example for some area
http://www.cafeconleche.org/books/xmljava/chapters/index.html

Java API for XML Processing (JAXP) 1.4

JAXP has SIX Major parts
1. SAX, 2. DOM, 3. TrAX, 4. XPath, 5. Validation, 6. StAX

Packages	
javax.xml	Defines core XML constants and functionality from the XML specifications.
javax.xml.datatype	XML/Java Type Mappings.
javax.xml.namespace	XML Namespace processing.
javax.xml.parsers	Provides classes allowing the processing of XML documents.
javax.xml.stream	
javax.xml.stream.events	
javax.xml.stream.util	
javax.xml.transform	This package defines the generic APIs for processing transformation instructions, and performing a transformation from source to result.
javax.xml.transform.dom	This package implements DOM-specific transformation APIs.
javax.xml.transform.sax	This package implements SAX2-specific transformation APIs.
javax.xml.transform.stax	Provides for StAX-specific transformation APIs.
javax.xml.transform.stream	This package implements stream- and URI- specific transformation APIs.
javax.xml.validation	This package provides an API for validation of XML documents.

since JAXP 1.3

some SAX example
http://www.java-tips.org/java-se-tips/org.xml.sax/

next round, please set the factory SYSTEM property in every example

it is Introduced by SUN

1st round - SUN Material
2nd round - Deleted some of page
3rd round - deleted sun material itself
finished on 22 - Nov - 08
4th round - 24 Nov 2008

1 11
2 3
3 5
4 2
5 2

it has 8 methods.

2 - Element related
2 - Document related
2 - Prefix related
1 - element text
1 - Locator

29 - Nov - 08

org.xml.sax

Interface ContentHandler

All Known Subinterfaces:
TemplatesHandler, TransformerHandler

so, the same interface can be used for Schema and DTD

All Known Implementing Classes:
DefaultHandler, ~~DefaultHandler2~~, ValidatorHandler, ~~XMLFilterImpl~~, ~~XMLReaderAdapter~~

`public interface` **`ContentHandler`**

~~Receive notification of the logical content of a document.~~

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* ***NO WARRANTY****. See http://www.saxproject.org for further information.*~~

~~This is the main interface that most SAX applications implement: if the application needs to be informed of basic parsing events, it implements this interface and registers an instance with the SAX parser using the `setContentHandler` method. The parser uses the instance to report basic document-related events like the start and end of elements and character data.~~

~~The order of events in this interface is very important, and mirrors the order of information in the document itself. For example, all of an element's content (character data, processing instructions, and/or subelements) will appear, in order, between the startElement event and the corresponding endElement event.~~

~~This interface is similar to the now-deprecated SAX 1.0 DocumentHandler interface, but it adds support for Namespaces and for reporting skipped entities (in non-validating XML processors).~~

~~Implementors should note that there is also a `ContentHandler` class in the `java.net` package; that means that it's probably a bad idea to do~~

```
import java.net.*;
 import org.xml.sax.*;
```

In fact, "import ...*" is usually a sign of sloppy programming anyway, so the user should consider this a feature rather than a bug.

Since:
SAX 2.0
Version:
2.0.1+ (sax2r3pre1)
Author:
David Megginson
See Also:
`XMLReader`, `DTDHandler`, `ErrorHandler`

Method Summary

void	**characters**(char[] ch, int start, int length) Receive notification of character data.
void	**endDocument**() Receive notification of the end of a document.
void	**endElement**(java.lang.String uri, java.lang.String localName, java.lang.String qName) Receive notification of the end of an element.
void	**endPrefixMapping**(java.lang.String prefix) End the scope of a prefix-URI mapping.
void	**ignorableWhitespace**(char[] ch, int start, int length) Receive notification of ignorable whitespace in element content. only with DTD
void	**processingInstruction**(java.lang.String target, java.lang.String data) Receive notification of a processing instruction.
void	**setDocumentLocator**(Locator locator) Receive an object for locating the origin of SAX document events.

void	~~skippedEntity(java.lang.String name)~~ only with DTD Receive notification of a skipped entity.
void	startDocument() ~~Receive notification of the beginning of a document.~~
void	startElement(java.lang.String uri, java.lang.String localName, java.lang.String qName, Attributes ~~atts)~~ ~~Receive notification of the beginning of an element.~~
void	startPrefixMapping(java.lang.String prefix, java.lang.String uri) ~~Begin the scope of a prefix-URI Namespace mapping.~~

Method Detail

setDocumentLocator

this Locator object has not overridden version of toString() method as StAX Locator

void **setDocumentLocator**(Locator locator)

~~Receive an object for locating the origin of SAX document events.~~

SAX parsers are strongly encouraged (though not absolutely required) to supply a locator: if it does so, it must supply the locator to the application by invoking this method before invoking any of the other methods in the ContentHandler interface.

~~The locator allows the application to determine the end position of any document-related event, even if the parser is not reporting an error. Typically, the application will use this information for reporting its own errors (such as character content that does not match an application's business rules). The information returned by the locator is probably not sufficient for use with a search engine.~~

Note that the locator will return correct information only during the invocation SAX event callbacks after startDocument returns and before endDocument is called. The application should not attempt to use it at any other time.

~~**Parameters:**~~

~~locator - an object that can return the location of any SAX document event~~

~~**See Also:**~~

Locator

---

startDocument

```
void startDocument()
                    throws SAXException
```

~~Receive notification of the beginning of a document.~~

The SAX parser will invoke this method only once, before any other event callbacks (except for setDocumentLocator).

i have verified. this method called before to all event call back methods.

~~**Throws:**~~

SAXException ~~- any SAX exception, possibly wrapping another exception~~

~~**See Also:**~~

endDocument()

---

endDocument

```
void endDocument()
                  throws SAXException
```

~~Receive notification of the end of a document.~~

There is an apparent contradiction between the documentation for this method and the documentation for ErrorHandler.fatalError(org.xml.sax.SAXParseException). Until this ambiguity is resolved in a future major release, clients should make no assumptions about whether endDocument() will or will not be invoked when the parser has reported a fatalError() or thrown an exception.

~~The SAX parser will inv~~ ~~ast method invoked during the parse. The parser sha~~ ~~andoned parsing (because of an unrecoverable erro~~

So, Never believe this method will be executed at every time.
so, releasing resource has to be done somewhere.

~~**Throws:**~~

`SAXException` - any SAX exception, possibly wrapping another exception

See Also:

`startDocument()`

---

startPrefixMapping

```
void startPrefixMapping(java.lang.String prefix,
                        java.lang.String uri)
                        throws SAXException
```

the Namespace MUST be enabled on FACTORY instance, these event callback method has to be called.

Begin the scope of a prefix-URI Namespace mapping.

The information from this event is not necessary for normal Namespace processing: the SAX XML reader will automatically replace prefixes for element and attribute names when the `http://xml.org/sax/features/namespaces` feature is *true* (the default).

USE

There are cases, however, when applications need to use prefixes in character data or in attribute values, where they cannot safely be expanded automatically; the start/endPrefixMapping event supplies the information to the application to expand prefixes in those contexts itself, if necessary.

Note that start/endPrefixMapping events are not guaranteed to be properly nested relative to each other: all startPrefixMapping events will occur immediately before the corresponding `startElement` event, and all `endPrefixMapping` events will occur immediately after the corresponding `endElement` event, but their order is not otherwise guaranteed.

There should never be start/endPrefixMapping events for the "xml" prefix, since it is predeclared and immutable.

Parameters:

`prefix` - the Namespace prefix being declared. An empty string is used for the default element namespace, which has no prefix.

`uri` - the Namespace URI the prefix is mapped to

Throws:

`SAXException` - the client may throw an exception during processing

See Also:

`endPrefixMapping(java.lang.String)`, `startElement(java.lang.String, java.lang.String, java.lang.String, org.xml.sax.Attributes)`

---

endPrefixMapping

```
void endPrefixMapping(java.lang.String prefix)
                      throws SAXException
```

~~End the scope of a prefix-URI mapping.~~

~~See~~ `startPrefixMapping` ~~for details. These events will always occur immediately after the corresponding~~ `endElement` ~~event, but the order of~~ `endPrefixMapping` ~~events is not otherwise guaranteed.~~

~~**Parameters:**~~

`prefix` ~~- the prefix that was being mapped. This is the empty string when a default mapping scope ends.~~

~~**Throws:**~~

`SAXException` ~~- the client may throw an exception during processing~~

~~**See Also:**~~

~~`startPrefixMapping(java.lang.String, java.lang.String)`, `endElement(java.lang.String, java.lang.String, java.lang.String)`~~

---

startElement

these **uri, localName** are null if namespace processing is not enabled on FACTORY instance

```
void startElement(java.lang.String uri,
                  java.lang.String localName,
                  java.lang.String qName,
                  Attributes atts)
                  throws SAXException
```

Receive notification of the beginning of an element.

~~The Parser will invoke this method at the beginning of every element in the XML document; there will be a corresponding~~ `endElement` ~~event for every startElement event (even when the element is empty). All of the element's content will be reported, in order, before the~~

~~corresponding endElement event.~~

~~This event allows up to three name components for each element:~~

1. ~~the Namespace URI;~~
2. ~~the local name; and~~
3. ~~the qualified (prefixed) name.~~

Any or all of these may be provided, depending on the values of the *http://xml.org/sax/features/namespaces* and the *http://xml.org/sax/features/namespace-prefixes* properties:

- the Namespace URI and local name are required when the namespaces property is *true* (the default), and are optional when the namespaces property is *false* (if one is specified, both must be);
- the qualified name is required when the namespace-prefixes property is *true*, and is optional when the namespace-prefixes property is *false* (the default).

~~Note that the attribute list provided will contain only attributes with explicit values (specified or defaulted): #IMPLIED attributes will be omitted. The attribute list will contain attributes used for Namespace declarations (xmlns* attributes) only if the~~ `http://xml.org/sax/features/namespace-prefixes` ~~property is true (it is false by default, and support for a true value is optional).~~

~~Like~~ `characters()`, ~~attribute values may have characters that need more than one~~ `char` ~~value.~~

Parameters:

`uri` ~~- the Namespace URI, or the empty string if the element has no Namespace URI or if Namespace processing is not being performed~~

`localName` ~~- the local name (without prefix), or the empty string if Namespace processing is not being performed~~

`qName` ~~- the qualified name (with prefix), or the empty string if qualified names are not available~~

`atts` ~~- the attributes attached to the element. If there are no attributes, it shall be an empty Attributes object. The value of this object after startElement returns is undefined~~

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

See Also:

`endElement(java.lang.String, java.lang.String, java.lang.String)`, `Attributes`, `AttributesImpl`

---

endElement

```
void endElement(java.lang.String uri,
                java.lang.String localName,
                java.lang.String qName)
                throws SAXException
```

~~Receive notification of the end of an element.~~

~~The SAX parser will invoke this method at the end of every element in the XML document; there will be a corresponding `startElement` event for every endElement event (even when the element is empty).~~

~~For information on the names, see startElement.~~

~~**Parameters:**~~

~~`uri` - the Namespace URI, or the empty string if the element has no Namespace URI or if Namespace processing is not being performed~~

~~`localName` - the local name (without prefix), or the empty string if Namespace processing is not being performed~~

~~`qName` - the qualified XML name (with prefix), or the empty string if qualified names are not available~~

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

---

characters

```
void characters(char[] ch,
                int start,
                int length)
                throws SAXException
```

In actuality, the `ch` character array includes the entire document. The application must not attempt to read characters outside the range the event feeds to the `characters()` event.

DO:

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; ~~however,~~

~~all of the characters in any single event must come from the same external entity so that the Locator provides useful information.~~

~~The application must not attempt to read from the array outside of the specified range.~~

~~Individual characters may consist of more than one Java `char` value. There are two important cases where this happens, because characters can't be represented in just sixteen bits. In one case, characters are represented in a *Surrogate Pair*, using two special Unicode values. Such characters are in the so-called "Astral Planes", with a code point above U+FFFF. A second case involves composite characters, such as a base character combining with one or more accent characters.~~

~~Your code should not assume that algorithms using `char`-at-a-time idioms will be working in character units; in some cases they will split characters. This is relevant wherever XML permits arbitrary characters, such as attribute values, processing instruction data, and comments as well as in data reported from this method. It's also generally relevant whenever Java code manipulates internationalized text; the issue isn't unique to XML.~~

~~Note that some parsers will report whitespace in element content using the `ignorableWhitespace` method rather than this one (validating parsers *must* do so).~~

~~Parameters:~~

~~`ch` - the characters from the XML document~~

~~`start` - the start position in the array~~

~~`length` - the number of characters to read from the array~~

~~Throws:~~

~~`SAXException` - any SAX exception, possibly wrapping another exception~~

~~See Also:~~

~~`ignorableWhitespace(char[], int, int)`, `Locator`~~

~~ignorableWhitespace~~

```
void ignorableWhitespace(char[] ch,
                         int start,
                         int length)
                         throws SAXException
```

this will be work only when a xml document associated with DTD

~~Receive notification of ignorable whitespace in element content.~~

Validating Parsers must use this method to report each chunk of whitespace in element content (see the W3C XML 1.0 recommendation, section 2.10): non-validating parsers may also use this method if they are capable of parsing and using content models.

SAX parsers may return all contiguous whitespace in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

The application must not attempt to read from the array outside of the specified range.

Parameters:

`ch` - the characters from the XML document

`start` - the start position in the array

`length` - the number of characters to read from the array

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

See Also:

`characters(char[], int, int)`

---

processingInstruction

```
void processingInstruction(java.lang.String target,
                           java.lang.String data)
                           throws SAXException
```

Receive notification of a processing instruction.

The Parser will invoke this method once for each processing instruction found: note that processing instructions may occur before or after the main document element.

A SAX parser must never report an XML declaration (XML 1.0, section 2.8) or a text declaration (XML 1.0, section 4.3.1) using this method.

Like `characters()`, processing instruction data may have characters that need more than one `char` value.

Parameters:

`target` - the processing instruction target

`data` - the processing instruction data, or null if none was supplied. The data does not include any whitespace separating it from the target

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

---

skippedEntity

it will be work only with DTD

```
void skippedEntity(java.lang.String name)
                   throws SAXException
```

Receive notification of a skipped entity. This is not called for entity references within markup constructs such as element start tags or markup declarations. (The XML recommendation requires reporting skipped external entities. SAX also reports internal entity expansion/non-expansion, except within markup constructs.)

The Parser will invoke this method each time the entity is skipped. Non-validating processors may skip entities if they have not seen the declarations (because, for example, the entity was declared in an external DTD subset). All processors may skip external entities, depending on the values of the `http://xml.org/sax/features/external-general-entities` and the `http://xml.org/sax/features/external-parameter-entities` properties.

Parameters:

`name` - the name of the skipped entity. If it is a parameter entity, the name will begin with '%', and if it is the external DTD subset, it will be the string "[dtd]"

Throws:

`SAXException` - any SAX exception, possibly wrapping another exception

---

29 - Nov - 08

it don't have any method as its own

Just Adaptor for handler interfaces are

1. ErrorHandler
2. ContentHandler

org.xml.sax.helpers

Class DefaultHandler

```
java.lang.Object
  └org.xml.sax.helpers.DefaultHandler
```

All Implemented Interfaces:

ContentHandler, ~~DTDHandler~~, ~~EntityResolver~~, ErrorHandler

Direct Known Subclasses:

DefaultHandler2

```
public class DefaultHandler
```

All the method has only empty implementation

extends java.lang.Object
implements EntityResolver, DTDHandler, ContentHandler, ErrorHandler

~~Default base class for SAX2 event handlers.~~

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with **NO WARRANTY**.* See http://www.saxproject.org for further information.~~

~~This class is available as a convenience base class for SAX2 applications: it provides default implementations for all of the callbacks in the four core SAX2 handler classes:~~

- `EntityResolver`
- `DTDHandler`
- `ContentHandler`
- `ErrorHandler`

~~Application writers can extend this class when they need to implement only part of an interface; parser writers can instantiate this class to provide default handlers when the application has not supplied its~~

~~own.~~

~~This class replaces the deprecated SAX1~~ `HandlerBase` ~~class~~.

~~Since:~~

~~SAX 2.0~~

~~Version:~~

~~2.0.1 (sax2r2)~~

~~Author:~~

~~David Megginson,~~

~~See Also:~~

`EntityResolver`, `DTDHandler`, `ContentHandler`, `ErrorHandler`

~~Constructor Summary~~

`DefaultHandler``()`

Method Summary

void	**`characters`**`(char[] ch, int start, int length)` Receive notification of character data inside an element.
void	**`endDocument`**`()` Receive notification of the end of the document.
void	**`endElement`**`(java.lang.String uri, java.lang.String localName, java.lang.String qName)` Receive notification of the end of an element.
void	**`endPrefixMapping`**`(java.lang.String prefix)` Receive notification of the end of a Namespace mapping.
void	**`error`**`(SAXParseException e)` Receive notification of a recoverable parser error.
void	**`fatalError`**`(SAXParseException e)` Report a fatal XML parsing error.

void	**ignorableWhitespace**(char[] ch, int start, int length) Receive notification of ignorable whitespace in element content.
void	**notationDecl**(java.lang.String name, java.lang.String publicId, java.lang.String systemId) Receive notification of a notation declaration.
void	**processingInstruction**(java.lang.String target, java.lang.String data) Receive notification of a processing instruction.
InputSource	**resolveEntity**(java.lang.String publicId, java.lang.String systemId) Resolve an external entity.
void	**setDocumentLocator**(Locator locator) Receive a Locator object for document events.
void	**skippedEntity**(java.lang.String name) Receive notification of a skipped entity.
void	**startDocument**() Receive notification of the beginning of the document.
void	**startElement**(java.lang.String uri, java.lang.String localName, java.lang.String qName, Attributes attributes) Receive notification of the start of an element.
void	**startPrefixMapping**(java.lang.String prefix, java.lang.String uri) Receive notification of the start of a Namespace mapping.
void	**unparsedEntityDecl**(java.lang.String name, java.lang.String publicId, java.lang.String systemId, java.lang.String notationName) Receive notification of an unparsed entity declaration.
void	**warning**(SAXParseException e) Receive notification of a parser warning.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

10 - Dec - 08

it has 2 methods

org.xml.sax

Interface ErrorHandler

All Known Implementing Classes:

DefaultHandler, DefaultHandler2, HandlerBase, XMLFilterImpl

`public interface` **`ErrorHandler`**

this is only XMLReader can be set ErrorHandler. nowhere else.

~~Basic interface for SAX error handlers.~~

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with **NO WARRANTY**.* See http://www.saxproject.org for further information.~~

If a SAX application needs to implement customized error handling, it must implement this interface and then register an instance with the XML reader using the `setErrorHandler` method. ~~The parser will then report all errors and warnings through this interface.~~

WARNING: If an application does *not* register an ErrorHandler, XML parsing errors will go unreported, except that *SAXParseException*s will be thrown for fatal errors. In order to detect validity errors, an ErrorHandler that does something with `error()` calls must be registered.

~~For XML processing errors, a SAX driver must use this interface in preference to throwing an exception: it is up to the application to decide whether to throw an exception for different types of errors and warnings. Note, however, that there is no requirement that the parser continue to report additional errors after a call to~~ `fatalError` ~~In other words, a SAX driver class may throw an exception after reporting any fatalError. Also parsers may throw appropriate exceptions for non-XML errors. For example,~~ `XMLReader.parse()` ~~would throw an IOException for errors accessing entities or the document.~~

~~**Since:**~~

~~SAX 1.0~~

~~**Version:**~~

~~2.0.1+ (sax2r3pre1)~~

Author:

David Megginson

See Also:

XMLReader.setErrorHandler(org.xml.sax.ErrorHandler), SAXParseException

---

Method Summary

void	**error**(SAXParseException exception) Receive notification of a recoverable error.
void	**fatalError**(SAXParseException exception) Receive notification of a non-recoverable error.
void	**warning**(SAXParseException exception) Receive notification of a warning.

this is used to report schema validation error.

this method used to report non well formed element errors

Method Detail

warning

```
void warning(SAXParseException exception)
             throws SAXException
```

This method usefull only with DTD

Receive notification of a warning.

SAX parsers will use this method to report conditions that are not errors or fatal errors as defined by the XML recommendation. The default behaviour is to take no action.

The SAX parser must continue to provide normal parsing events after invoking this method: it should still be possible for the application to process the document through to the end.

Filters may use this method to report other, non-XML warnings as well.

Parameters:

`exception` - The warning information encapsulated in a SAX parse exception.

Throws:

~~SAXException - Any SAX exception, possibly wrapping another exception.~~

~~**See Also:**~~

~~SAXParseException~~

---

error

```
void error(SAXParseException exception)
           throws SAXException
```

Receive notification of a recoverable error.

~~This corresponds to the definition of "error" in section 1.2 of the W3C XML 1.0 Recommendation. For example, a validating parser would use this callback to report the violation of a validity constraint. The default behaviour is to take no action.~~

~~The SAX parser must continue to provide normal parsing events after invoking this method: it should still be possible for the application to process the document through to the end. If the application cannot do so, then the parser should report a fatal error even if the XML recommendation does not require it to do so.~~

? Filters may use this method to report other, non-XML errors as well. ?

how to use filter ?? need sample code

Parameters:

`exception` - The error information encapsulated in a SAX parse exception.

Throws:

SAXException - Any SAX exception, possibly wrapping another exception.

See Also:

SAXParseException

---

fatalError

```
void fatalError(SAXParseException exception)
                throws SAXException
```

Receive notification of a non-recoverable error.

There is an apparent contradiction between the documentation for this method and the documentation for `ContentHandler.endDocument()`. Until this ambiguity is resolved in a future major release, clients should make no assumptions about whether endDocument () will or will not be invoked when the parser has reported a fatalError() or thrown an exception.

This corresponds to the definition of "fatal error" in section 1.2 of the W3C XML 1.0 Recommendation. For example, a parser would use this callback to report the violation of a well-formedness constraint.

The application must assume that the document is unusable after the parser has invoked this method, and should continue (if at all) only for the sake of collecting additional error messages: in fact, SAX parsers are free to stop reporting any other events once this method has been invoked.

~~**Parameters:**~~

~~`exception` - The error information encapsulated in a SAX parse exception.~~

~~**Throws:**~~

~~`SAXException` - Any SAX exception, possibly wrapping another exception.~~

~~**See Also:**~~

`SAXParseException`

Pac .sax

it is only for reading a xml document. DOM has to be used to update

just learn 7 interfaces, 4 exceptions

1. Content Handler
2. ErrorHandler

This package provides the core SAX APIs.

See:

Description

SAX used Observer Design pattern

Interface Summary

~~AttributeList~~	~~**Deprecated.** *This interface has been replaced by the SAX2* `Attributes` *interface, which includes Namespace support.*~~
Attributes	Interface for a list of XML attributes. 1
ContentHandler	Receive notification of the logical content of a document. 2
~~DocumentHandler~~	~~**Deprecated.** *This interface has been replaced by the SAX2* `ContentHandler` *interface, which includes Namespace support.*~~
~~DTDHandler~~	~~Receive notification of basic DTD-related events.~~
~~EntityResolver~~	~~Basic interface for resolving entities.~~
ErrorHandler	Basic interface for SAX error handlers. 3
Locator	Interface for associating a SAX event with a document location. 4
~~Parser~~	~~**Deprecated.** *This interface has been replaced by the SAX2* `XMLReader` *interface, which includes Namespace support.*~~
XMLFilter	Interface for an XML filter. 5
XMLReader	Interface for reading an XML document using callbacks. 6

Class Summary

~~HandlerBase~~	~~**Deprecated.** *This class works with the deprecated* `DocumentHandler` *interface.*~~
InputSource	A single input source for an XML entity. 7

Exception Summary	
SAXException	Encapsulate a general SAX error or warning. 8
SAXNotRecognizedException	Exception class for an unrecognized identifier. 9
SAXNotSupportedException	Exception class for an unsupported operation. 10
SAXParseException	Encapsulate an XML parse error or warning. 11

Package org.xml.sax Description

~~This package provides the core SAX APIs. Some SAX1 APIs are deprecated to encourage integration of namespace-awareness into designs of new applications and into maintenance of existing infrastructure.~~

~~See http://www.saxproject.org for more information about SAX.~~

SAX2 Standard Feature Flags

for example,
url + Feature id = will key for feature
http://xml.org/sax/features/external-general-entities

One of the essential characteristics of SAX2 is that it added feature flags which can be used to examine and perhaps modify parser modes, in particular modes such as validation. Since features are identified by (absolute) URIs, anyone can define such features. Currently defined standard feature URIs have the prefix `http://xml.org/sax/features/` before an identifier such as `validation`. Turn features on or off using *setFeature*. Those standard identifiers are:

Feature ID	Access	Default	Description
external-general-entities 1	*read/write*	*unspecified*	Reports whether this parser processes external general entities; always true if validating.
external-parameter-entities 2	*read/write*	*unspecified*	Reports whether this parser processes external parameter entities; always true if validating.
is-standalone 3	(parsing) *read-only*, (not parsing) *none*	not applicable	May be examined only during a parse, after the *startDocument()* callback has been completed; read-only. The value is true if the document specified standalone="yes" in its XML declaration, and otherwise is false.

lexical-handler/ parameter-entities 4	*read/write*	*unspecified*	A value of "true" indicates that the LexicalHandler will report the beginning and end of parameter entities.
namespaces 5	*read/write*	true	A value of "true" indicates namespace URIs and unprefixed local names for element and attribute names will be available.
namespace-prefixes 6	*read/write*	false	A value of "true" indicates that XML qualified names (with prefixes) and attributes (including *xmlns** attributes) will be available.
resolve-dtd-uris 7	*read/write*	*true*	A value of "true" indicates that system IDs in declarations will be absolutized (relative to their base URIs) before reporting. (That is the default behavior for all SAX2 XML parsers.) A value of "false" indicates those IDs will not be absolutized; parsers will provide the base URI from *Locator.getSystemId()*. This applies to system IDs passed in • *DTDHandler.notationDecl()*, • *DTDHandler.unparsedEntityDecl()*, and • *DeclHandler.externalEntityDecl()*. It does not apply to *EntityResolver.resolveEntity()*, which is not used to report declarations, or to *LexicalHandler.startDTD()*, which already provides the non-absolutized URI.
string-interning 8	*read/write*	*unspecified*	Has a value of "true" if all XML names (for elements, prefixes, attributes, entities, notations, and local names), as well as Namespace URIs, will have been interned using *java.lang.String.intern*. This supports fast testing of equality/inequality against string constants, rather than forcing slower calls to *String.equals()*.

unicode-normalization-checking 9	*read/write*	*false*	Controls whether the parser reports Unicode normalization errors as described in section 2.13 and Appendix B of the XML 1.1 Recommendation. If true, Unicode normalization errors are reported using the ErrorHandler.error() callback. Such errors are not fatal in themselves (though, obviously, other Unicode-related encoding errors may be).
use-attributes2 10	*read-only*	not applicable	Returns "true" if the *Attributes* objects passed by this parser in *ContentHandler. startElement()* implement the *org.xml.sax.ext. Attributes2* interface. That interface exposes additional DTD-related information, such as whether the attribute was specified in the source text rather than defaulted.
use-locator2 11	*read-only*	not applicable	Returns "true" if the *Locator* objects passed by this parser in *ContentHandler. setDocumentLocator()* implement the *org. xml.sax.ext.Locator2* interface. That interface exposes additional entity information, such as the character encoding and XML version used.
use-entity-resolver2 12	*read/write*	*true*	Returns "true" if, when *setEntityResolver* is given an object implementing the *org.xml. sax.ext.EntityResolver2* interface, those new methods will be used. Returns "false" to indicate that those methods will not be used.
validation 13	*read/write*	*unspecified*	Controls whether the parser is reporting all validity errors; if true, all external entities will be read.

xmlns-uris 14	*read/write*	*false*	Controls whether, when the *namespace-prefixes* feature is set, the parser treats namespace declaration attributes as being in the *http://www.w3.org/2000/xmlns/* namespace. By default, SAX2 conforms to the original "Namespaces in XML" Recommendation, which explicitly states that such attributes are not in any namespace. Setting this optional flag to "true" makes the SAX2 events conform to a later backwards-incompatible revision of that recommendation, placing those attributes in a namespace.
xml-1.1 15	*read-only*	not applicable	Returns "true" if the parser supports both XML 1.1 and XML 1.0. Returns "false" if the parser supports only XML 1.0.

Support for the default values of the *namespaces* and *namespace-prefixes* properties is required. Support for any other feature flags is entirely optional.

For default values not specified by SAX2, each XMLReader implementation specifies its default, or may choose not to expose the feature flag. Unless otherwise specified here, implementations may support changing current values of these standard feature flags, but not while parsing.

SAX2 Standard Handler and Property IDs

For parser interface characteristics that are described as objects, a separate namespace is defined. The objects in this namespace are again identified by URI, and the standard property URIs have the prefix `http://xml.org/sax/properties/` before an identifier such as `lexical-handler` or `dom-node`. Manage those properties using *setProperty()*. Those identifiers are:

Property ID	Description
declaration-handler	Used to see most DTD declarations except those treated as lexical ("document element name is ...") or which are mandatory for all SAX parsers (*DTDHandler*). The Object must implement *org.xml.sax.ext.DeclHandler*.
document-xml-version	May be examined only during a parse, after the startDocument() callback has been completed; read-only. This property is a literal string describing the actual XML version of the document, such as "1.0" or "1.1".

dom-node	For "DOM Walker" style parsers, which ignore their *parser.parse()* parameters, this is used to specify the DOM (sub)tree being walked by the parser. The Object must implement the *org.w3c.dom.Node* interface.
lexical-handler	Used to see some syntax events that are essential in some applications: comments, CDATA delimiters, selected general entity inclusions, and the start and end of the DTD (and declaration of document element name). The Object must implement *org.xml.sax.ext.LexicalHandler*.
xml-string	Readable only during a parser callback, this exposes a **TBS** chunk of characters responsible for the current event.

All of these standard properties are optional; XMLReader implementations need not support them.

Package org.xml.sax.ext

nothing is need from this package except lexical handler

This package contains interfaces to SAX2 facilities that conformant SAX drivers won't necessarily support.

See:

Description

Interface Summary	
~~Attributes2~~	~~SAX2 extension to augment the per-attribute information provided though~~ `Attributes`
~~DeclHandler~~	~~SAX2 extension handler for DTD declaration events.~~
~~EntityResolver2~~	~~Extended interface for mapping external~~ entity references to input sources, or providing a missing external subset.
LexicalHandler	SAX2 extension handler for lexical events.
~~Locator2~~	~~SAX2 extension to augment the entity information provided though a~~ `Locator`

Class Summary	
~~Attributes2Impl~~	~~SAX2 extension helper for additional Attributes information, implementing the~~ `Attributes2` ~~interface.~~
~~DefaultHandler2~~	~~This class extends the SAX2 base handler class to support the SAX2~~ `LexicalHandler`, `DeclHandler`, ~~and~~ `EntityResolver2` ~~extensions.~~
~~Locator2Impl~~	~~SAX2 extension helper for holding additional Entity information, implementing the~~ `Locator2` ~~interface.~~

Package org.xml.sax.ext Description

This package contains interfaces to SAX2 facilities that conformant SAX drivers won't necessarily support.

See http://www.saxproject.org for more information about SAX.

This package is independent of the SAX2 core, though the functionality exposed generally needs to be implemented within a parser core. That independence has several consequences:

- SAX2 drivers are *not* required to recognize these handlers.
- You cannot assume that the class files will be present in every SAX2 installation.
- This package may be updated independently of SAX2 (i.e. new handlers and classes may be added without updating SAX2 itself).
- The new handlers are not implemented by the SAX2 `org.xml.sax.helpers.DefaultHandler` or `org.xml.sax.helpers.XMLFilterImpl` classes. You can subclass these if you need such behavior, or use the helper classes found here.
- The handlers need to be registered differently than core SAX2 handlers.

This package, SAX2-ext, is a standardized extension to SAX2. It is designed both to allow SAX parsers to pass certain types of information to applications, and to serve as a simple model for other SAX2 parser extension packages. Not all such extension packages should need to be recognized directly by parsers, however. As an example, most validation systems can be cleanly layered on top of parsers supporting the standardized SAX2 interfaces.

Package org.xml.sax.helpers

This package contains "helper" classes, including support for bootstrapping SAX-based applications.

See:

need only 3 classes

Description

Class Summary	
~~AttributeListImpl~~	~~**Deprecated.** *This class implements a deprecated interface, `AttributeList`; that interface has been replaced by `Attributes`, which is implemented in the `AttributesImpl` helper class.*~~
~~AttributesImpl~~	~~Default implementation of the Attributes interface.~~
DefaultHandler	Default base class for SAX2 event handlers.
~~LocatorImpl~~	~~Provide an optional convenience implementation of Locator.~~
~~NamespaceSupport~~	~~Encapsulate Namespace logic for use by applications using SAX, or internally by SAX drivers.~~
ParserAdapter	~~Adapt a SAX1 Parser as a SAX2 XMLReader.~~
~~ParserFactory~~	~~**Deprecated.** *This class works with the deprecated `Parser` interface.*~~
XMLFilterImpl	Base class for deriving an XML filter.
~~XMLReaderAdapter~~	~~Adapt a SAX2 XMLReader as a SAX1 Parser.~~
XMLReaderFactory	Factory for creating an XML reader.

Package org.xml.sax.helpers Description

This package contains "helper" classes, including support for bootstrapping SAX-based applications.

See http://www.saxproject.org for more information about SAX.

10 - Dec - 08

this is NOT TRAX Source

5 - setter methods
5 - getter methods

4 - constructors

for example, if we set all character stream, byte stream, uri , then parser will give use only character stream. it just ignore byte stream and uri

org.xml.sax

Class InputSource

```
java.lang.Object
  └org.xml.sax.InputSource
```

set the SystemID useful irrespective of any stream, will help to proper error reporting.

```
public class InputSource
```

1. char stream
2. byte stream
3. URI connection

extends java.lang.Object

use this, for any Frameworked JAXP coding

A single input source for an XML entity.

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* ***NO WARRANTY***. See http://www.saxproject.org for further information.~~

This class allows a SAX application to encapsulate information about an input source in a single object, which may include a public identifier, a system identifier, a byte stream (possibly with a specified encoding), and/or a character stream.

There are two places that the application can deliver an input source to the parser: as the argument to the Parser.parse method, or as the ~~return value of the EntityResolver.resolveEntity method.~~

The SAX parser will use the InputSource object to determine how to read XML input. If there is a character stream available, the parser will read that stream directly, disregarding any text encoding declaration found in that stream. If there is no character stream, but there is a byte stream, the parser will use that byte stream, using the encoding specified in the InputSource or else (if no encoding is specified) autodetecting the character encoding using an algorithm such as the one in the XML specification. If neither a character stream nor a byte stream is available, the parser will attempt to open a URI connection to the resource identified by the system identifier.

An InputSource object belongs to the application: the SAX parser shall never modify it in any way (it may modify a copy if necessary). However, standard processing of both byte and character streams is to close them on as part of end-of-parse cleanup, so applications should not attempt to re-use such streams

after they have been handed to a parser.

~~Since:~~

~~SAX 1.0~~

~~Version:~~

~~2.0.1 (sax2r2)~~

~~Author:~~

~~David Megginson~~

~~See Also:~~

~~XMLReader.parse(org.xml.sax.InputSource), EntityResolver.resolveEntity(java.lang.String, java.lang.String), InputStream, Reader~~

Constructor Summary

InputSource() Zero-argument default constructor.
InputSource(java.io.InputStream byteStream) Create a new input source with a byte stream.
InputSource(java.io.Reader characterStream) Create a new input source with a character stream.
InputSource(java.lang.String systemId) Create a new input source with a system identifier.

Method Summary

java.io.InputStream	**getByteStream**() Get the byte stream for this input source.
java.io.Reader	**getCharacterStream**() Get the character stream for this input source.
java.lang.String	**getEncoding**() Get the character encoding for a byte stream or URI.
java.lang.String	**getPublicId**() Get the public identifier for this input source.

java.lang.String	**getSystemId**() Get the system identifier for this input source.
void	**setByteStream**(java.io.InputStream byteStream) Set the byte stream for this input source.
void	**setCharacterStream**(java.io.Reader characterStream) Set the character stream for this input source.
void	**setEncoding**(java.lang.String encoding) Set the character encoding, if known.
void	**setPublicId**(java.lang.String publicId) Set the public identifier for this input source.
void	**setSystemId**(java.lang.String systemId) Set the system identifier for this input source.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

InputSource

public **InputSource**()

Zero-argument default constructor.

See Also:
setPublicId(java.lang.String), setSystemId(java.lang.String), setByteStream(java.io.InputStream), setCharacterStream(java.io.Reader), setEncoding(java.lang.String)

InputSource

`public InputSource(java.lang.String systemId)`

Create a new input source with a system identifier.

Applications may use setPublicId to include a public identifier as well, or setEncoding to specify the character encoding, if known.

If the system identifier is a URL, it must be fully resolved (it may not be a relative URL).

Parameters:

`systemId` - The system identifier (URI).

See Also:

`setPublicId(java.lang.String)`, `setSystemId(java.lang.String)`, `setByteStream(java.io.InputStream)`, `setEncoding(java.lang.String)`, `setCharacterStream(java.io.Reader)`

InputSource

`public InputSource(java.io.InputStream byteStream)`

Create a new input source with a byte stream.

Application writers should use setSystemId() to provide a base for resolving relative URIs, may use setPublicId to include a public identifier, and may use setEncoding to specify the object's character encoding.

Parameters:

`byteStream` - The raw byte stream containing the document.

See Also:

`setPublicId(java.lang.String)`, `setSystemId(java.lang.String)`, `setEncoding(java.lang.String)`, `setByteStream(java.io.InputStream)`, `setCharacterStream(java.io.Reader)`

InputSource

```
public InputSource(java.io.Reader characterStream)
```

Create a new input source with a character stream.

Application writers should use setSystemId() to provide a base for resolving relative URIs, and may use setPublicId to include a public identifier.

The character stream shall not include a byte order mark.

See Also:
`setPublicId(java.lang.String)`, `setSystemId(java.lang.String)`, `setByteStream(java.io.InputStream)`, `setCharacterStream(java.io.Reader)`

Method Detail

setPublicId

```
public void setPublicId(java.lang.String publicId)
```

~~Set the public identifier for this input source.~~

~~The public identifier is always optional: if the application writer includes one, it will be provided as part of the location information.~~

~~**Parameters:**~~
`publicId` ~~- The public identifier as a string.~~
~~**See Also:**~~
`getPublicId()`, `Locator.getPublicId()`, `SAXParseException.getPublicId()`

---

getPublicId

```
public java.lang.String getPublicId()
```

~~Get the public identifier for this input source.~~

~~**Returns:**~~

~~The public identifier, or null if none was supplied.~~

~~**See Also:**~~

`setPublicId(java.lang.String)`

---

setSystemId

```
public void setSystemId(java.lang.String systemId)
```

Set the system identifier for this input source.

The system identifier is optional if there is a byte stream or a character stream, but it is still useful to provide one, since the application can use it to resolve relative URIs and can include it in error messages and warnings (the parser will attempt to open a connection to the URI only if there is no byte stream or character stream specified).

If the application knows the character encoding of the object pointed to by the system identifier, it can register the encoding using the setEncoding method.

If the system identifier is a URL, it must be fully resolved (it may not be a relative URL).

Parameters:

`systemId` - The system identifier as a string.

See Also:

`setEncoding(java.lang.String)`, `getSystemId()`, `Locator.getSystemId()`, `SAXParseException.getSystemId()`

---

getSystemId

```
public java.lang.String getSystemId()
```

Get the system identifier for this input source.

The getEncoding method will return the character encoding of the object pointed to, or null if unknown.

If the system ID is a URL, it will be fully resolved.

Returns:
The system identifier, or null if none was supplied.
See Also:
`setSystemId(java.lang.String)`, `getEncoding()`

---

setByteStream

```
public void setByteStream(java.io.InputStream byteStream)
```

so, even if we set byteStream, if we set charStream also, it will be ignored.

Set the byte stream 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.

Parameters:
`byteStream` - A byte stream containing an XML document or other entity.
See Also:
`setEncoding(java.lang.String)`, `getByteStream()`, `getEncoding()`, `InputStream`

---

getByteStream

```
public java.io.InputStream getByteStream()
```

Get the byte stream for this input source.

The getEncoding method will return the character encoding for this byte stream, or null if unknown.

Returns:
The byte stream, or null if none was supplied.

See Also:
`getEncoding()`, `setByteStream(java.io.InputStream)`

---

setEncoding

so, encoding will come to effect only if it is ByteStream

```
public void setEncoding(java.lang.String encoding)
```

Set the character encoding, if known.

The encoding must be a string acceptable for an XML encoding declaration (see section 4.3.3 of the XML 1.0 recommendation).

This method has no effect when the application provides a character stream.

Parameters:
`encoding` - A string describing the character encoding.

See Also:
`setSystemId(java.lang.String)`, `setByteStream(java.io.InputStream)`, `getEncoding()`

---

getEncoding

```
public java.lang.String getEncoding()
```

Get the character encoding for a byte stream or URI. This value will be ignored when the application provides a character stream.

Returns:
The encoding, or null if none was supplied.

See Also:
`setByteStream(java.io.InputStream)`, `getSystemId()`, `getByteStream()`

---

setCharacterStream

`public void **setCharacterStream**(java.io.Reader characterStream)`

Set the character stream 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.

Parameters:

`characterStream` - The character stream containing the XML document or other entity.

See Also:

`getCharacterStream()`, `Reader`

---

getCharacterStream

`public java.io.Reader **getCharacterStream**()`

Get the character stream for this input source.

Returns:

The character stream, or null if none was supplied.

See Also:

`setCharacterStream(java.io.Reader)`

```
xmlReader.setProperty("http://xml.org/sax/properties/
lexical-handler", this);
```

10 - Dec - 08

org.xml.sax.ext

Interface LexicalHandler

There are 3 methods.

2 - CDATA
1 - comment

All Known Subinterfaces:

TransformerHandler

this handler only can be set in either
SAXParser or XMLReader
(i tested)

All Known Implementing Classes:

DefaultHandler2

`public interface` **`LexicalHandler`**

~~SAX2 extension handler for lexical events.~~

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* ***NO WARRANTY****. See http://www.saxproject.org for further information.*~~

This is an optional extension handler for SAX2 to provide lexical information about an XML document, such as comments and CDATA section boundaries. XML readers are not required to recognize this handler, and it is not part of core-only SAX2 distributions.

SO, comment / CDATA can be any where in XML Document

The events in the lexical handler apply to the entire document, not just to the document element, and all lexical handler events must appear between the content handler's startDocument and endDocument events.

also in SAXReader (i tested)

To set the LexicalHandler for an XML reader, use the `setProperty` method with the property name `http://xml.org/sax/properties/lexical-handler` and an object implementing this interface (or null) as the value. If the reader does not report lexical events, it will throw a `SAXNotRecognizedException` when you attempt to register the handler.

Since:

SAX 2.0 (extensions 1.0)

Version:

2.0.1 (sax2r2)

Author:

David Megginson

Method Summary

void	**comment**(char[] ch, int start, int length) Report an XML comment anywhere in the document.
void	**endCDATA**() Report the end of a CDATA section.
~~void~~	~~**endDTD**()~~ ~~Report the end of DTD declarations.~~
~~void~~	~~**endEntity**(java.lang.String name)~~ ~~Report the end of an entity.~~
void	**startCDATA**() Report the start of a CDATA section.
~~void~~	~~**startDTD**(java.lang.String name, java.lang.String publicId, java.lang.String systemId)~~ ~~Report the start of DTD declarations, if any.~~
~~void~~	~~**startEntity**(java.lang.String name)~~ ~~Report the beginning of some internal and external XML entities.~~

Method Detail

~~startDTD~~

```
void startDTD(java.lang.String name,
              java.lang.String publicId,
              java.lang.String systemId)
              throws SAXException
```

~~Report the start of DTD declarations, if any.~~

~~This method is intended to report the beginning of the DOCTYPE declaration; if the document has no DOCTYPE declaration, this method will not be invoked.~~

~~All declarations reported through `DTDHandler` or `DeclHandler` events must appear between the startDTD and `endDTD` events. Declarations are assumed to belong to the internal DTD subset unless they appear between `startEntity` and `endEntity` events. Comments and processing instructions from the DTD should also be reported between the startDTD and endDTD events, in their original order of (logical) occurrence; they are not required to appear in their correct locations relative to DTDHandler or DeclHandler events, however.~~

~~Note that the start/endDTD events will appear within the start/endDocument events from ContentHandler and before the first `startElement` event.~~

~~**Parameters:**~~

~~`name` - The document type name.~~

~~`publicId` - The declared public identifier for the external DTD subset, or null if none was declared.~~

~~`systemId` - The declared system identifier for the external DTD subset, or null if none was declared. (Note that this is not resolved against the document base URI.)~~

~~**Throws:**~~

~~`SAXException` - The application may raise an exception.~~

~~**See Also:**~~

~~`endDTD()`, `startEntity(java.lang.String)`~~

~~endDTD~~

~~`void` **`endDTD()`**~~
~~`throws SAXException`~~

~~Report the end of DTD declarations.~~

~~This method is intended to report the end of the DOCTYPE declaration; if the document has no DOCTYPE declaration, this method will not be invoked.~~

~~**Throws:**~~

~~`SAXException` - The application may raise an exception.~~

~~**See Also:**~~

startDTD(java.lang.String, java.lang.String, java.lang.String)

~~startEntity~~

~~void **startEntity**(java.lang.String name)~~
~~throws SAXException~~

~~Report the beginning of some internal and external XML entities.~~

~~The reporting of parameter entities (including the external DTD subset) is optional, and SAX2 drivers that report LexicalHandler events may not implement it; you can use the `http://xml.org/sax/features/lexical-handler/parameter-entities` feature to query or control the reporting of parameter entities.~~

~~General entities are reported with their regular names, parameter entities have '%' prepended to their names, and the external DTD subset has the pseudo-entity name "[dtd]".~~

~~When a SAX2 driver is providing these events, all other events must be properly nested within start/end entity events. There is no additional requirement that events from `DeclHandler` or `DTDHandler` be properly ordered.~~

~~Note that skipped entities will be reported through the `skippedEntity` event, which is part of the ContentHandler interface.~~

~~Because of the streaming event model that SAX uses, some entity boundaries cannot be reported under any circumstances:~~

- ~~general entities within attribute values~~
- ~~parameter entities within declarations~~

~~These will be silently expanded, with no indication of where the original entity boundaries were.~~

~~Note also that the boundaries of character references (which are not really entities anyway) are not reported.~~

~~All start/endEntity events must be properly nested.~~

Parameters:

`name` - The name of the entity. If it is a parameter entity, the name will begin with '%', and if it is the external DTD subset, it will be "[dtd]".

Throws:

`SAXException` - The application may raise an exception.

See Also:

`endEntity(java.lang.String)`, `DeclHandler.internalEntityDecl(java.lang.String, java.lang.String)`, `DeclHandler.externalEntityDecl(java.lang.String, java.lang.String, java.lang.String)`

---

endEntity

```
void endEntity(java.lang.String name)
               throws SAXException
```

Report the end of an entity.

Parameters:

`name` - The name of the entity that is ending.

Throws:

`SAXException` - The application may raise an exception.

See Also:

`startEntity(java.lang.String)`

---

startCDATA

```
void startCDATA()
               throws SAXException
```

Report the start of a CDATA section.

The contents of the CDATA section will be reported through the regular `characters` event; this event is intended only to report the boundary.

~~**Throws:**~~

SAXException ~~- The application may raise an exception.~~

~~**See Also:**~~

endCDATA()

---

endCDATA

```
void endCDATA()
               throws SAXException
```

Report the end of a CDATA section.

Throws:

SAXException - The application may raise an exception.

See Also:

startCDATA()

---

comment

```
void comment(char[] ch,
             int start,
             int length)
             throws SAXException
```

Report an XML comment anywhere in the document.

This callback will be used for comments inside or outside the document element, including comments in the external DTD subset (if read). ~~Comments in the DTD must be properly nested inside start/endDTD and start/endEntity events (if used).~~

~~**Parameters:**~~

ch ~~- An array holding the characters in the comment.~~

start ~~- The starting position in the array.~~

length ~~- The number of characters to use from the array.~~

~~**Throws:**~~

SAXException ~~- The application may raise an exception.~~

11 -Dec - 08

org.xml.sax

Interface Locator

only 4 methods

All Known Subinterfaces:

Locator2

All Known Implementing Classes:

Locator2Impl, LocatorImpl

it is not like Location of StAX,

The implementation of Locator is not override its toString() method

---

`public interface` **`Locator`**

Interface for associating a SAX event with a document location.

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* ***NO WARRANTY****. See http://www.saxproject.org for further information.*~~

~~If a SAX parser provides location information to the SAX application, it does so by implementing this interface and then passing an instance to the application using the content handler's `setDocumentLocator` method. The application can use the object to obtain the location of any other SAX event in the XML source document.~~

~~Note that the results returned by the object will be valid only during the scope of each callback method: the application will receive unpredictable results if it attempts to use the locator at any other time, or after parsing completes.~~

SAX parsers are not required to supply a locator, but they are very strongly encouraged to do so. If the parser supplies a locator, it must do so before reporting any other document events. ~~If no locator has been set by the time the application receives the `startDocument` event, the application should assume that a locator is not available.~~

~~**Since:**~~

~~SAX 1.0~~

~~**Version:**~~

~~2.0.1 (sax2r2)~~

~~**Author:**~~

~~David Megginson~~

~~**See Also:**~~

`ContentHandler.setDocumentLocator(org.xml.sax.Locator)`

Method Summary

`int`	**`getColumnNumber`**`()` ~~Return the column number where the current document event ends.~~
`int`	**`getLineNumber`**`()` Return ~~the line number where the current document event ends.~~
`java.lang.String`	**`getPublicId`**`()` ~~Return the public identifier for the current document event.~~ will have value only if the xml document comes over internet
`java.lang.String`	**`getSystemId`**`()` ~~Return the system identifier for the current document event.~~ it will have value only if the xml document comes from local machine hard disk

Method Detail

getPublicId

`java.lang.String` **`getPublicId`**`()`

Return the public identifier for the current document event.

~~The return value is the public identifier of the document entity or of the external parsed entity in which the markup triggering the event appears.~~

~~**Returns:**~~

~~A string containing the public identifier, or null if none is available.~~

~~**See Also:**~~

`getSystemId()`

---

getSystemId

`java.lang.String` **`getSystemId`**`()`

~~Return the system identifier for the current document event.~~

~~The return value is the system identifier of the document entity or of the external parsed entity in which the markup triggering the event appears.~~

~~If the system identifier is a URL, the parser must resolve it fully before passing it to the application. For example, a file name must always be provided as a *file:...* URL, and other kinds of relative URI are also resolved against their bases.~~

~~**Returns:**~~

~~A string containing the system identifier, or null if none is available.~~

~~**See Also:**~~

`getPublicId()`

---

getLineNumber

`int` **`getLineNumber`**`()`

~~Return the line number where the current document event ends. Lines are delimited by line ends, which are defined in the XML specification.~~

~~**Warning:** The return value from the method is intended only as an approximation for the sake of diagnostics; it is not intended to provide sufficient information to edit the character content of the original XML document. In some cases, these "line" numbers match what would be displayed as columns, and in others they may not match the source text due to internal entity expansion.~~

~~The return value is an approximation of the line number in the document entity or external parsed entity where the markup triggering the event appears.~~

~~If possible, the SAX driver should provide the line position of the first character after the text associated with the document event. The first line is line 1.~~

Returns:

The line number, or -1 if none is available.

See Also:

`getColumnNumber()`

---

getColumnNumber

`int` **`getColumnNumber`**`()`

Return the column number where the current document event ends. This is one-based number of Java `char` values since the last line end.

~~**Warning:** The return value from the method is intended only as an approximation for the sake of diagnostics; it is not intended to provide sufficient information to edit the character content of the original XML document. For example, when lines contain combining character sequences, wide characters, surrogate pairs, or bi-directional text, the value may not correspond to the column in a text editor's display.~~

~~The return value is an approximation of the column number in the document entity or external parsed entity where the markup triggering the event appears.~~

~~If possible, the SAX driver should provide the line position of the first character after the text associated with the document event. The first column in each line is column 1.~~

Returns:

The column number, or -1 if none is available.

See Also:

`getLineNumber()`

---

if you have separate handler class for different kind of event like error, parse event, lexical event use directly with XMLReader

if not use SAXParser with DefaultHandler adaptor class

28 - Nov - 08

javax.xml.parsers

Class SAXParse

it has setter / getter method ONLY for Property but not for Feature

XMLReader has both

14 methods

5 - parse methods
2 - get /set property
3 - Feature GET methods (only .. no setter)
1 - Schema related
1 - get XMLReader
1 - DEPRECATED

```
java.lang.Object
  └ javax.xml.parsers.SAXParser
```

```
public abstract class SAXParser
```

extends java.lang.Object

Defines the API that wraps an `XMLReader` implementation class. In JAXP 1.0, this class wrapped the `Parser` interface, however this interface was replaced by the `XMLReader`. For ease of transition, this class continues to support the same name and interface as well as supporting new methods. An instance of this class can be obtained from the `SAXParserFactory.newSAXParser()` method. Once an instance of this class is obtained, XML can be parsed from a variety of input sources. These input sources are InputStreams, Files, URLs, and SAX InputSources.

~~This static method creates a new factory instance based on a system property setting or uses the platform default if no property has been defined.~~

~~The system property that controls which Factory implementation to create is named `"javax.xml.parsers.SAXParserFactory"`. This property names a class that is a concrete subclass of this abstract class. If no property is defined, a platform default will be used.~~

~~As the content is parsed by the underlying parser, methods of the given `HandlerBase` or the `DefaultHandler` are called.~~

~~Implementors of this class which wrap an underlaying implementation can consider using the `ParserAdapter` class to initially adapt their SAX1 implementation to work under this revised class.~~

Version:

$Revision: 1.6 $, $Date: 2007/01/27 01:26:27 $

Author:
Jeff Suttor

input for SAX Parser can be either

1. File
2. InputSource
3. InputStream
4. String (file Path)

BUT not Reader

Constructor Summary

protected	**SAXParser**() Protected constructor to prevent instaniation.

Method Summary

~~abstract Parser~~	~~**getParser**()~~ this is deprecated.. Replaced by XMLReader ~~Returns the SAX parser that is encapsultated by the implementation of this class.~~
abstract java.lang.Object	**getProperty**(java.lang.String name) Returns the particular property requested for in the underlying implementation of XMLReader.
~~Schema~~	~~**getSchema**()~~ ~~Get a reference to the the Schema being used by the XML processor.~~
~~abstract XMLReader~~	~~**getXMLReader**()~~ ~~Returns the XMLReader that is encapsulated by the implementation of this class.~~
1 abstract boolean	**isNamespaceAware**() getter for FEATURE ~~Indicates whether or not this parser is configured to understand namespaces.~~
2 abstract boolean	**isValidating**() getter for FEATURE ~~Indicates whether or not this parser is configured to validate XML documents.~~
3 boolean	**isXIncludeAware**() getter for FEATURE ~~Get the XInclude processing mode for this parser.~~
1 void	**parse**(java.io.File f, DefaultHandler dh) ~~Parse the content of the file specified as XML using the specified DefaultHandler.~~
~~void~~	~~**parse**(java.io.File f, HandlerBase hb)~~ deprecated in SAX 2.0 ~~ontent of the file specified as XML using the specified HandlerBase.~~

2 void	**parse**(InputSource is, DefaultHandler dh) Parse the content give use, this InputSource for integrated and ARCHITECTURAL FRAMEWORK of JAXP coding. instead of using other overloaded parse method of SAXParser, this inputsource make easy for code maintain and support any kind of xml input. see that class's javadoc DefaultHandler
void deprecated in SAX 2.0	**parse**(InputSource i Parse the content given InputSource as XML using the specified HandlerBase
3 void	**parse**(java.io.InputStream is, DefaultHandler dh) Parse the content of the given InputStream instance as XML using the specified DefaultHandler
4 void	**parse**(java.io.InputStream is, DefaultHandler dh, java.lang.String systemId) ok. general input stream Parse the content of the given InputStream instance as XML using the specified DefaultHand not ENCODING parameter like StAX
void deprecated in SAX 2.0	**parse**(java.io.InputStream is, HandlerBase hb) e content of the given InputStream instance as XML using the specified HandlerBase.
void deprecated in SAX 2.0	**parse**(java.io.InputStream is, HandlerBase hb, java.lang.String systemId) e content of the given InputStream instance as XML using the specified HandlerBase.
5 void	**parse**(java.lang.String uri, DefaultHandler dh) Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified DefaultHandler
void deprecated in SAX 2.0	**parse**(java.lang.String uri, HandlerBase hb) se the content described by the giving Uniform Resource Identifier (URI) as XML using the specified HandlerBase
void	**reset**() Reset this SAXParser to its original configuration.
abstract void	**setProperty**(java.lang.String name, java.lang.Object value) Sets the particular property in the underlying implementation of XMLReader

Methods inherited from class java.lang.Object

~~clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait~~

Constructor Detail

SAXParser

~~protected **SAXParser()**~~

~~Protected constructor to prevent instaniation. Use SAXParserFactory.newSAXParser()~~

Method Detail

reset

SAXParser is not thread-safe, so it cannot be used more than one thread at a time

`public void **reset**()`

Reset this `SAXParser` to its original configuration.

`SAXParser` is reset to the same state as when it was created with `SAXParserFactory.newSAXParser()`. `reset()` is designed to allow the reuse of existing `SAXParser`s thus saving resources associated with the creation of new `SAXParser`s.

The reset `SAXParser` is not guaranteed to have the same `Schema` `Object`, e.g. `Object.equals(Object obj)`. It is guaranteed to have a functionally equal `Schema`.

~~**Throws:**~~

~~`java.lang.UnsupportedOperationException` - When Implementations do not override this method~~

~~**Since:**~~

~~1.5~~

---

parse

deprecated in SAX 2.0

```
public void parse(java.io.InputStream is,
                  HandlerBase hb)
          throws SAXException,
                 java.io.IOException
```

deprecated

Parse the content of the given `InputStream` instance as XML using the specified `HandlerBase`. *Use of the DefaultHandler version of this method is recommended as the HandlerBase class has been deprecated in SAX 2.0.*

Parameters:

`is` - InputStream containing the content to be parsed.

`hb` - The SAX HandlerBase to use.

Throws:

`java.lang.IllegalArgumentException` - If the given InputStream is null.

`SAXException` - If parse produces a SAX error.

`java.io.IOException` - If an IO error occurs interacting with the `InputStream`.

See Also:

`DocumentHandler`

---

parse

deprecated in SAX 2.0

```
public void parse(java.io.InputStream is,
                  HandlerBase hb,
                  java.lang.String systemId)
          throws SAXException,
                 java.io.IOException
```

deprecated

Parse the content of the given `InputStream` instance as XML using the specified `HandlerBase`. *Use of the DefaultHandler version of this method is recommended as the HandlerBase class has been deprecated in SAX 2.0.*

Parameters:

`is` - InputStream containing the content to be parsed.

`hb` - The SAX HandlerBase to use.

`systemId` - The systemId which is needed for resolving relative URIs.

Throws:

`java.lang.IllegalArgumentException` - If the given `InputStream` is `null`.

`java.io.IOException` - If any IO error occurs interacting with the `InputStream`.
`SAXException` - If any SAX errors occur during processing.

See Also:
`version of this method instead.`

---

parse

```
public void parse(java.io.InputStream is,
                  DefaultHandler dh)
           throws SAXException,
                  java.io.IOException
```

~~Parse the content of the given `InputStream` instance as XML using the specified `DefaultHandler`.~~

Parameters:
`is` - InputStream containing the content to be parsed.
`dh` - The SAX DefaultHandler to use.

Throws:
`java.lang.IllegalArgumentException` - If the given InputStream is null.
`java.io.IOException` - If any IO errors occur.
`SAXException` - If any SAX errors occur during processing.

See Also:
`DocumentHandler`

---

parse

actually, this system id will give full path of file, and parser will append internal hierarchy of element of WITHIN xml file when reporting any error

```
public void parse(java.io.InputStream is,
                  DefaultHandler dh,
                  java.lang.String systemId)
           throws SAXException,
                  java.io.IOException
```

~~Parse the content of the given `InputStream` instance as XML using the specified `DefaultHandler`.~~

Parameters:

`is` - InputStream containing the content to be parsed.

`dh` - The SAX DefaultHandler to use.

`systemId` - The systemId which is needed for resolving relative URIs.

Throws:

`java.lang.IllegalArgumentException` - If the given InputStream is null.

`java.io.IOException` - If any IO errors occur.

`SAXException` - If any SAX errors occur during processing.

See Also:

`version of this method instead.`

---

parse

deprecated

```
public void parse(java.lang.String uri,
                  HandlerBase hb)
           throws SAXException,
                  java.io.IOException
```

Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified `HandlerBase`. *Use of the DefaultHandler version of this method is recommended as the `HandlerBase` class has been deprecated in SAX 2.0*

Parameters:

`uri` - The location of the content to be parsed.

`hb` - The SAX HandlerBase to use.

Throws:

`java.lang.IllegalArgumentException` - If the uri is null.

`java.io.IOException` - If any IO errors occur.

`SAXException` - If any SAX errors occur during processing.

See Also:

`DocumentHandler`

---

parse

```
public void parse(java.lang.String uri,
```

```
                DefaultHandler dh)
        throws SAXException,
                java.io.IOException
```

~~Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified~~ `DefaultHandler`

~~**Parameters:**~~

~~`uri` - The location of the content to be parsed.~~

~~`dh` - The SAX DefaultHandler to use.~~

~~**Throws:**~~

~~`java.lang.IllegalArgumentException` - If the uri is null.~~

~~`java.io.IOException` - If any IO errors occur.~~

~~`SAXException` - If any SAX errors occur during processing.~~

~~**See Also:**~~

~~`DocumentHandler`~~

~~**parse**~~

deprecated

```
public void parse(java.io.File f,
                HandlerBase hb)
        throws SAXException,
                java.io.IOException
```

~~Parse the content of the file specified as XML using the specified~~ `HandlerBase`. ~~*Use of the DefaultHandler version of this method is recommended as the HandlerBase class has been deprecated in SAX 2.0*~~

~~**Parameters:**~~

~~`f` - The file containing the XML to parse~~

~~`hb` - The SAX HandlerBase to use.~~

~~**Throws:**~~

~~`java.lang.IllegalArgumentException` - If the File object is null.~~

~~`java.io.IOException` - If any IO errors occur.~~

~~`SAXException` - If any SAX errors occur during processing.~~

See Also:

`DocumentHandler`

---

parse

```
public void parse(java.io.File f,
                  DefaultHandler dh)
           throws SAXException,
                  java.io.IOException
```

Parse the content of the file specified as XML using the specified `DefaultHandler`.

Parameters:

`f` - The file containing the XML to parse

`dh` - The SAX DefaultHandler to use.

Throws:

`java.lang.IllegalArgumentException` - If the File object is null.

`java.io.IOException` - If any IO errors occur.

`SAXException` - If any SAX errors occur during processing.

See Also:

`DocumentHandler`

---

parse

deprecated

```
public void parse(InputSource is,
                  HandlerBase hb)
           throws SAXException,
                  java.io.IOException
```

Parse the content given `InputSource` as XML using the specified `HandlerBase`. *Use of the DefaultHandler version of this method is recommended as the HandlerBase class has been deprecated in SAX 2.0*

Parameters:

`is` - The InputSource containing the content to be parsed.

`hb` - The SAX HandlerBase to use.

Throws:

`java.lang.IllegalArgumentException` - If the `InputSource` object is

null

java.io.IOException - If any IO errors occur.

SAXException - If any SAX errors occur during processing.

See Also:

DocumentHandler

---

parse

this InputSource is NOT same used in TrAX.
SAXSource, StreamSource are used in TrAX

```
public void parse(InputSource is,
                  DefaultHandler dh)
           throws SAXException,
                  java.io.IOException
```

Parse the content given InputSource as XML using the specified DefaultHandler.

Parameters:

is - The InputSource containing the content to be parsed.

dh - The SAX DefaultHandler to use.

Throws:

java.lang.IllegalArgumentException - If the InputSource object is null

java.io.IOException - If any IO errors occur.

SAXException - If any SAX errors occur during processing.

See Also:

DocumentHandler

---

getParser

deprecated

```
public abstract Parser getParser()
                       throws SAXException
```

Returns the SAX parser that is encapsultated by the implementation of this class.

Returns:

The SAX parser that is encapsultated by the implementation of this class.

~~**Throws:**~~

~~SAXException - If any SAX errors occur during processing.~~

---

getXMLReader

```
public abstract XMLReader getXMLReader()
                                throws SAXException
```

Returns the XMLReader that is encapsulated by the implementation of this class.

~~**Returns:**~~

~~The XMLReader that is encapsulated by the implementation of this class.~~

~~**Throws:**~~

~~SAXException - If any SAX errors occur during processing.~~

---

isNamespaceAware

Q1

```
public abstract boolean isNamespaceAware()
```

~~Indicates whether or not this parser is configured to understand namespaces.~~

~~**Returns:**~~

~~true if this parser is configured to understand namespaces; false otherwise.~~

---

isValidating

Q2

```
public abstract boolean isValidating()
```

~~Indicates whether or not this parser is configured to validate XML documents.~~

~~**Returns:**~~

~~true if this parser is configured to validate XML documents; false otherwise.~~

---

setProperty

```
public abstract void setProperty(java.lang.String name,
                                 java.lang.Object value)
                          throws SAXNotRecognizedException,
                                 SAXNotSupportedException
```

Sets the particular property in the underlying implementation of `XMLReader`. A list of the core features and properties can be found at http://sax.sourceforge.net/?selected=get-set.

~~**Parameters:**~~

`name` ~~- The name of t~~

`value` ~~- The value of~~

Property name cannot be any arbitrary name.
if it is, will throw SAXNotRecognizedException

~~**Throws:**~~

`SAXNotRecognizedException` ~~- When the underlying XMLReader does not recognize the property name.~~

`SAXNotSupportedException` ~~- When the underlying XMLReader recognizes the property name but doesn't support the property.~~

~~**See Also:**~~

`XMLReader.setProperty(java.lang.String, java.lang.Object)`

---

getProperty

```
public abstract java.lang.Object getProperty(java.lang.String name)
                                 throws SAXNotRecognizedException,
                                        SAXNotSupportedException
```

~~Returns the particular property requested for in the underlying implementation of~~ `XMLReader`.

~~**Parameters:**~~

`name` ~~- The name of the property to be retrieved.~~

~~**Returns:**~~

~~Value of the requested property~~.

Throws:

`SAXNotRecognizedException` - When the underlying XMLReader does not recognize the property name.

`SAXNotSupportedException` - When the underlying XMLReader recognizes the

property name but doesn't support the property.

See Also:

`XMLReader.getProperty(java.lang.String)`

---

getSchema

Q3

`public Schema getSchema()`

~~Get a reference to the the `Schema` being used by the XML processor.~~

~~If no schema is being used, `null` is returned.~~

~~**Returns:**~~

`Schema` ~~being used or~~ `null` ~~if none in use~~

~~**Throws:**~~

`java.lang.UnsupportedOperationException` ~~- When implementation does not override this method~~

~~**Since:**~~

~~1.5~~

---

isXIncludeAware

Q4

`public boolean isXIncludeAware()`

Get the XInclude processing mode for this parser.

~~**Returns:**~~

~~the return value of the~~ `SAXParserFactory.isXIncludeAware()` ~~when this parser was created from factory.~~

~~**Throws:**~~

`java.lang.UnsupportedOperationException` ~~- When implementation does not override this method~~

~~**Since:**~~

~~1.5~~

~~**See Also:**~~

`SAXParserFactory.setXIncludeAware(boolean)`

Here after, just evaluate only IMPORTANT methods not just like setter / getter methods (18 - June - 09)

26 - Nov - 08

13 Methods

2 - Schema related
2 - feature related
2 - creating instance
2 - namespace aware
2 - validation related
2 - XInclude related
1 - getting SAX Parser

javax.xml.parsers

Class SAXParserFactory

```
java.lang.Object
  └javax.xml.parsers.SAXParserFactory
```

Schema object can be set only in DocumentBuilderFactory for DOM. Nowhere else !!!

```
public abstract class SAXParserFactory
```

extends java.lang.Object

it has setter / getter method ONLY for feature but not for property

XMLReader has both

~~Defines a factory API that enables applications to configure and obtain a SAX based parser to parse XML documents.~~

Version:
$Revision: 1.6 $,

Author:
Jeff Suttor, Neera

this is one of best example, where we have to use Abstract class and Interface to achieve OOPS abstraction.

if we need any origin use Abstract class with static methods or Constructor.
if we need 100 % abstraction use java interface

Constructor Summary

protected	**SAXParserFactory**() Protected constructor to force use of newInstance().

Method Summary

abstract boolean	**getFeature**(java.lang.String name) Returns the particular property requested for in the underlying implementation
Schema	**getSchema**() ~~Gets the S~~ ema schema) meth

All Factory method are start with "new / create" and appended by class name

newSAXParesr()
newValidator()
newValidatorHandler()
newDocumentBuilder()
newTransformer() and etc....

boolean	**isNamespaceAware**() ~~Indicates whether or not the factory is configured to produce parsers which are namespace aware.~~
boolean	**isValidating**() ~~Indicates whether or not the factory is configured to produce parsers which validate the XML content during parse.~~
boolean	**isXIncludeAware**() ~~Get state of XInclude processing.~~
static SAXParserFactory	**newInstance**() ~~Obtain a new instance of a SAXParserFactory~~
static SAXParserFactory	**newInstance**(java.lang.String factoryClassName, java.lang.ClassLoader classLoader) ~~Obtain a new instance of a SAXParserFactory from class name.~~
abstract SAXParser	**newSAXParser**() Creates a new instance of a SAXParser using the currently configured factory parameters.
abstract void	**setFeature**(java.lang.String name, boolean value) Sets the particular feature in the underlying implementation of org.xml.sax.XMLReader.
void	**setNamespaceAware**(boolean awareness) Specifies that the parser produced by this code will provide support for XML namespaces.
use of new style of ValidationFrame work of JAXP 1.3 void	**setSchema**(Schema schema) Set the Schema to be used by parsers created from this factory.
this is only place can set Schema instance in SAX API. nowhere else void	**setValidating**(boolean validating) Specifies that the parser produced by this code will validate documents as they are parsed.
void	**setXIncludeAware**(boolean state) Set state of XInclude processing. since JAXP 1.3

~~Methods inherited from class java.lang.Object~~
~~clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait~~

Constructor Detail

SAXParserFactory

`protected SAXParserFactory()`

~~Protected constructor to force use of~~ `newInstance()`.

Method Detail

newInstance

`public static SAXParserFactory newInstance()`

By default this is null

~~Obtain a new instance of a~~ `SAXParserFactory`. ~~This static method creates a new factory instance This method uses~~ the following ordered lookup procedure to determine the `SAXParserFactory` implementation class to load:

- Use the `javax.xml.parsers.SAXParserFactory` system property.
- Use the properties file "lib/jaxp.properties" in the JRE directory. This configuration file is in standard `java.util.Properties` format and contains the fully qualified name of the implementation class with the key being the system property defined above. The jaxp.properties file is read only once by the JAXP implementation and it's values are then cached for future use. If the file does not exist when the first attempt is made to read from it, no further attempts are made to check for its existence. It is not possible to change the value of any property in jaxp.properties after it has been read for the first time.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for a classname in the file `META-INF/services/javax.xml.parsers.SAXParserFactory` in jars available to the runtime.
- Platform default `SAXParserFactory` instance.

i could not find this property file in either JRE / JDK lib directory

~~Once an application has obtained a reference to a~~ `SAXParserFactory` ~~it can use the factory to configure and obtain parser instances.~~

this is the "com.sun.org.apache.xerces.internal.jaxp.SAXParserFactoryImpl" SAXParserFactory class used by Default in JRE 1.6 . Enable "jaxp.debug" system property to verify the actual factory class name

Tip for Trouble-shooting

Setting the `jaxp.debug` system property will cause this method to print a lot of debug messages to `System.err` about what it is doing and where it is looking at.

~~If you have problems loading~~ `DocumentBuilders`, ~~try:~~

```
java -Djaxp.debug=1 YourProgram ....
```

Printing mistake. here should be SAXParser

~~**Returns:**~~

~~A new instance of a SAXParserFactory.~~

Throws:

`FactoryConfigurationError` - if the implementation is not available or cannot be instantiated.

No need try-catch block

---

newInstance

```
public static SAXParserFactory newInstance(java.lang.String factoryClassName,
                                           java.lang.ClassLoader classLoader)
```

Obtain a new instance of a `SAXParserFactory` from class name. This function is useful when there are multiple providers in the classpath. It gives more control to the application as it can specify which provider should be loaded.

class loader can be NULL

~~Once an application has obtained a reference to a `SAXParserFactory` it can use the factory to configure and obtain parser instances.~~

Tip for Trouble-shooting

~~Setting the `jaxp.debug` system property will cause this method to print a lot of debug messages to `System.err` about what it is doing and where it is looking at.~~

~~If you have problems, try:~~

```
java -Djaxp.debug=1 YourProgram ....
```

Parameters:

`factoryClassName` - fully qualified factory class name that provides implementation of `javax.xml.parsers.SAXParserFactory`.

`classLoader` - `ClassLoader` used to load the factory class. If `null` current `Thread`'s context classLoader is used to load the factory class.

Returns:

New instance of a `SAXParserFactory`

~~**Throws:**~~

`FactoryConfigurationError` ~~- if `factoryClassName` is `null`, or the factory class cannot be loaded, instantiated.~~

~~**Since:**~~

~~1.6~~

~~**See Also:**~~

`newInstance()`

---

newSAXParser

```
public abstract SAXParser newSAXParser()
                                 throws ParserConfigurationException,
                                        SAXException
```

~~Creates a new instance of a SAXParser using the currently configured factory parameters~~.

Returns:
A new instance of a SAXParser.

Throws:
`ParserConfigurationException` - if a parser cannot be created which satisfies the requested configuration.
`SAXException` - for SAX errors.

---

setNamespaceAware

```
public void setNamespaceAware(boolean awareness)
```

Specifies that the parser produced by this code will provide support for XML namespaces. By default the value of this is set to `false`.

because of, The Namespace support will reduce the parser performance while parsing xml file, if need just enable.

Parameters:
`awareness` ~~- true if the parser produced by this code will provide support for XML namespaces; false otherwise.~~

---

setValidating

because of, the Validation support will reduce the parser performance while parsing xml file, if need just enable

```
public void setValidating(boolean validating)
```

Specifies that the parser produced by this code will validate documents as they are parsed. By default the value of this is set to `false`.

~~Note that "the validation" here means a validating parser as defined in the XML recommendation. In other words, it essentially just controls the DTD validation. (except the legacy two properties defined in JAXP 1.2.)~~

To use modern schema languages such as W3C XML Schema or RELAX NG instead of DTD, you can

configure your parser to be a non-validating parser by leaving the `setValidating(boolean)` method `false`, then use the `setSchema(Schema)` method to associate a schema to a parser.

use EITHER one,

1. enable validation feature on factory / parser instance
OR
2. set the Schema object on factory / parser instance

Parameters:

`validati` ... are parsed; false otherw

---

isNamespaceAware

```
public boolean isNamespaceAware()
```

Indicates whether or not the factory is configured to produce parsers which are namespace aware.

Returns:

true if the factory is configured to produce parsers which are namespace aware; false otherwise.

---

isValidating

```
public boolean isValidating()
```

Indicates whether or not the factory is configured to produce parsers which validate the XML content during parse.

Returns:

true if the factory is configured to produce parsers which validate the XML content during parse; false otherwise.

---

setFeature

```
public abstract void setFeature(java.lang.String name,
                                boolean value)
                         throws ParserConfigurationException,
                                SAXNotRecognizedException,
                                SAXNotSupportedException
```

Sets the particular feature in the underlying implementation of org.xml.sax.XMLReader. A list of the core features and properties can be found at http://www.saxproject.org/

All implementations are required to support the `XMLConstants.FEATURE_SECURE_PROCESSING` feature. When the feature is

super property, if u wanted controlled behavour of parsing

- `true`: the implementation will limit XML processing to conform to implementation limits. Examples include enity expansion limits and XML Schema constructs that would consume large amounts of resources. If XML processing is limited for security reasons, it will be reported via a call to the registered `ErrorHandler.fatalError(SAXParseException exception)`. See `SAXParser` `parse` methods for handler specification.
- When the feature is `false`, the implementation will processing XML according to the XML specifications without regard to possible implementation limits.

Super

~~Parameters:~~

`name` ~~- The name of the feature to be set.~~

`value` ~~- The value of the feature to be set.~~

Throws:

1 `ParserConfigurationException` - if a parser cannot be created which satisfies the requested configuration.

2 `SAXNotRecognizedException` - When the underlying XMLReader does not recognize the property name.

3 `SAXNotSupportedException` - When the underlying XMLReader recognizes the property name but doesn't support the property.

4 `java.lang.NullPointerException` - If the `name` parameter is null.

See Also:

`XMLReader.setFeature(java.lang.String, boolean)`

---

getFeature

```
public abstract boolean getFeature(java.lang.String name)
                        throws ParserConfigurationException,
                               SAXNotRecognizedException,
                               SAXNotSupportedException
```

~~Returns the particular property requested for in the underlying implementation of org.xml.sax.XMLReader.~~

~~Parameters:~~

`name` ~~- The name of the property to be retrieved.~~

~~Returns:~~

~~Value of the requested property~~.

~~Throws:~~

`ParserConfigurationException` ~~- if a parser cannot be created which satisfies the requested configuration~~.

~~SAXNotRecognizedException - When the underlying XMLReader does not recognize the property name.~~
~~SAXNotSupportedException - When the underlying XMLReader recognizes the property name but doesn't support the property.~~

~~**See Also:**~~
~~XMLReader.getProperty(java.lang.String)~~

---

getSchema

`public Schema getSchema()`

~~Gets the Schema object specified through the setSchema(Schema schema) method.~~

~~**Returns:**~~
~~the Schema object that was last set through the setSchema(Schema) method, or null if the method was not invoked since a SAXParserFactory is created.~~

~~**Throws:**~~
~~java.lang.UnsupportedOperationException - When implementation does not override this method~~

~~**Since:**~~
~~1.5~~

---

setSchema

`public void setSchema(Schema schema)`

Set the Schema to be used by parsers created from this factory.

~~When a Schema is non-null, a parser will use a validator created from it to validate documents before it passes information down to the application.~~

~~When warnings/errors/fatal errors are found by the validator, the parser must handle them as if those errors were found by the parser itself. In other words, if the user-specified ErrorHandler is set, it must receive those errors, and if not, they must be treated according to the implementation specific default error handling rules.~~

A validator may modify the SAX event stream (for example by adding default values that were missing in documents), and a parser is responsible to make sure that the application will receive those modified event stream.

Initialy, `null` is set as the `Schema`.

This processing will take effect even if the `isValidating()` method returns `false`.

It is an error to use the `http://java.sun.com/xml/jaxp/properties/schemaSource` property and/or the `http://java.sun.com/xml/jaxp/properties/schemaLanguage` property in conjunction with a non-null `Schema` object. Such configuration will cause a `SAXException` exception when those properties are set on a `SAXParser`.

so, setting schema properties on parser and set Schema object in factory are mutually exclusive

~~**Note for implmentors**~~

~~A parser must be able to work with any `Schema` implementation. However, parsers and schemas are allowed to use implementation-specific custom mechanisms as long as they yield the result described in the specification.~~

~~**Parameters:**~~

~~`schema` - `Schema` to use, `null` to remove a schema.~~

~~**Throws:**~~

~~`java.lang.UnsupportedOperationException` - When implementation does not override this method~~

~~**Since:**~~

~~1.5~~

---

setXIncludeAware

```
public void setXIncludeAware(boolean state)
```

Set state of XInclude processing.

~~If XInclude markup is found in the document instance, should it be processed as specified in XML Inclusions (XInclude) Version 1.0.~~

~~XInclude processing defaults to `false`.~~

~~**Parameters:**~~

~~`state` - Set XInclude processing to `true` or `false`~~

~~**Throws:**~~

~~`java.lang.UnsupportedOperationException` - When implementation does not override this method~~

~~**Since:**~~

1.5

---

isXIncludeAware

`public boolean` **`isXIncludeAware`**`()`

~~Get state of XInclude processing.~~

~~Returns:~~

~~current state of XInclude processing~~

~~Throws:~~

~~`java.lang.UnsupportedOperationException` When implementation does not override this method~~

~~Since:~~

~~1.5~~

---

11 -Dec - 08

org.xml.sax

Interface XMLFilter

only 2 methods as its own

All Superinterfaces:

XMLReader

see XMLFilterImpl class document for further information

All Known Implementing Classes:

XMLFilterImpl

`public interface` **`XMLFilter`**

extends XMLReader

Interface for an XML filter.

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* **NO WARRANTY**. See http://www.saxproject.org for further information.~~

An XML filter is like an XML reader, except that it obtains its events from another XML reader rather than a primary source like an XML document or database. Filters can modify a stream of events as they pass on to the final application.

super.. super

~~The XMLFilterImpl helper class provides a convenient base for creating SAX2 filters, by passing on all `EntityResolver`, `DTDHandler`, `ContentHandler` and `ErrorHandler` events automatically.~~

~~**Since:**~~

~~SAX 2.0~~

~~**Version:**~~

~~2.0.1 (sax2r2)~~

~~**Author:**~~

~~David Megginson~~

~~**See Also:**~~

~~`XMLFilterImpl`~~

Method Summary

XMLReader	**getParent**() Get the parent reader.
void	**setParent**(XMLReader parent) Set the parent reader.

Methods inherited from interface org.xml.sax.XMLReader
getContentHandler, getDTDHandler, getEntityResolver, getErrorHandler, getFeature, getProperty, parse, parse, setContentHandler, setDTDHandler, setEntityResolver, setErrorHandler, setFeature, setProperty

Method Detail

setParent

void **setParent**(XMLReader parent)

Set the parent reader.

This method allows the application to link the filter to a parent reader (which may be another filter). The argument may not be null.

Parameters:

parent - The parent reader.

---

getParent

XMLReader **getParent**()

Get the parent reader.

This method allows the application to query the parent reader (which may be another filter). It is generally a bad idea to perform any operations on the parent reader directly: they should all pass through this filter.

so, simply use parent to read the source document and let other task to filter

Returns:

The parent filter, or null if none has been set.

---

11 -Dec - 08

org.xml.sax.helpers

Class XMLFilterImpl

So, XMLFilter can be used to modify or skip the any SAXEvent.

it has 22 method.

But none of methods are new.
this class simply give empty implementation.

```
java.lang.Object
  └org.xml.sax.helpers.XMLFilterImpl
```

All Implemented Interfaces:

ContentHandler, ~~DTDHandler~~, ~~EntityResolver~~, ErrorHandler, XMLFilter, XMLReader

so, in our application we have to extend this class and override appropriate method TO MODIFY the EVENTS

the default implementation simply pass event to application

so, what are the real time requirement to modify the sax event ???

I DONT KNOW....

```
public c
```

extends java.lang.Object
implements XMLFilter, EntityResolver, DTDHandler, ContentHandler, ErrorHandler

~~Base class for deriving an XML filter.~~

i excuted a sample just to skip a particular event or pass the modified event into registered handler

~~*This module, both source code and documentation, is in the Public Domain, and comes with* **NO WARRANTY**. See http://www.saxproject.org for further information.~~

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.

set parent (XMLReader) , and input document into filter object

~~**Since:**~~

~~SAX 2.0~~

~~**Version:**~~

~~2.0.1 (sax2r2)~~

~~**Author:**~~

~~David Megginson~~

App's Content Handler

Our Extended XMLFilterImpl object to skip / modify the SAXEvent

XMLReader (as parent to filter object)

event flow

event flow

~~**See Also:**~~

~~`XMLFilter`, `XMLReader`, `EntityResolver`, `DTDHandler`, `ContentHandler`,~~

ErrorHandler

Constructor Summary

XMLFilterImpl() Construct an empty XML filter, with no parent.
XMLFilterImpl(XMLReader parent) Construct an XML filter with the specified parent.

Method Summary

void	**characters**(char[] ch, int start, int length) Filter a character data event.
void	**endDocument**() Filter an end document event.
void	**endElement**(java.lang.String uri, java.lang.String localName, java.lang.String qName) Filter an end element event.
void	**endPrefixMapping**(java.lang.String prefix) Filter an end Namespace prefix mapping event.
void	**error**(SAXParseException e) Filter an error event.
void	**fatalError**(SAXParseException e) Filter a fatal error event.
ContentHandler	**getContentHandler**() Get the content event handler.
DTDHandler	~~**getDTDHandler**()~~ ~~Get the current DTD event handler.~~
EntityResolver	~~**getEntityResolver**()~~ ~~Get the current entity resolver.~~
ErrorHandler	**getErrorHandler**() Get the current error event handler.

boolean	**getFeature**(java.lang.String name) Look up the value of a feature.
XMLReader	**getParent**() Get the parent reader.
java.lang.Object	**getProperty**(java.lang.String name) Look up the value of a property.
~~void~~	~~**ignorableWhitespace**(char[] ch, int start, int length)~~ ~~Filter an ignorable whitespace event.~~
~~void~~	~~**notationDecl**(java.lang.String name, java.lang.String publicId, java.lang.String systemId)~~ ~~Filter a notation declaration event.~~
void	**parse**(InputSource input) Parse a document.
void	**parse**(java.lang.String systemId) Parse a document.
~~void~~	~~**processingInstruction**(java.lang.String target, java.lang.String data)~~ ~~Filter a processing instruction event.~~
~~InputSource~~	~~**resolveEntity**(java.lang.String publicId, java.lang.String systemId)~~ ~~Filter an external entity resolution.~~
void	**setContentHandler**(ContentHandler handler) Set the content event handler.
void	**setDocumentLocator**(Locator locator) Filter a new document locator event.
~~void~~	~~**setDTDHandler**(DTDHandler handler)~~ ~~Set the DTD event handler.~~
~~void~~	~~**setEntityResolver**(EntityResolver resolver)~~ ~~Set the entity resolver.~~
void	**setErrorHandler**(ErrorHandler handler) Set the error event handler.

void	**setFeature**(java.lang.String name, boolean value) Set the value of a feature.
void	**setParent**(XMLReader parent) Set the parent reader.
void	**setProperty**(java.lang.String name, java.lang.Object value) Set the value of a property.
~~void~~	~~**skippedEntity**(java.lang.String name)~~ ~~Filter a skipped entity event.~~
void	**startDocument**() Filter a start document event.
void	**startElement**(java.lang.String uri, java.lang.String localName, java.lang.String qName, Attributes atts) Filter a start element event.
void	**startPrefixMapping**(java.lang.String prefix, java.lang.String uri) Filter a start Namespace prefix mapping event.
~~void~~	~~**unparsedEntityDecl**(java.lang.String name, java.lang.String publicId, java.lang.String systemId, java.lang.String notationName)~~ ~~Filter an unparsed entity declaration event.~~
~~void~~	~~**warning**(SAXParseException e)~~ ~~Filter a warning event.~~

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XMLFilterImpl

`public XMLFilterImpl()`

Construct an empty XML filter, with no parent.

This filter will have no parent: you must assign a parent before you start a parse or do any configuration with setFeature or setProperty, unless you use this as a pure event consumer rather than as an `XMLReader`.

See Also:
`XMLReader.setFeature(java.lang.String, boolean)`, `XMLReader.setProperty(java.lang.String, java.lang.Object)`, `setParent(org.xml.sax.XMLReader)`

XMLFilterImpl

`public XMLFilterImpl(XMLReader parent)`

Construct an XML filter with the specified parent.

See Also:
`setParent(org.xml.sax.XMLReader)`, `getParent()`

Method Detail

setParent

`public void setParent(XMLReader parent)`

Set the parent reader.

This is the `XMLReader` from which this filter will obtain its events and to which it will pass its configuration requests. The parent may itself be another filter.

If there is no parent reader set, any attempt to parse or to set or get a feature or property will fail.

Specified by:

setParent in interface XMLFilter

Parameters:

parent - The parent XML reader.

See Also:

getParent()

---

getParent

```
public XMLReader getParent()
```

Get the parent reader.

Specified by:

getParent in interface XMLFilter

Returns:

The parent XML reader, or null if none is set.

See Also:

setParent(org.xml.sax.XMLReader)

---

setFeature

```
public void setFeature(java.lang.String name,
                       boolean value)
                 throws SAXNotRecognizedException,
                        SAXNotSupportedException
```

Set the value of a feature.

This will always fail if the parent is null.

Specified by:

setFeature in interface XMLReader

Parameters:

name - The feature name.

value - The requested feature value.

Throws:

SAXNotRecognizedException - If the feature value can't be assigned or retrieved from the parent.
SAXNotSupportedException - When the parent recognizes the feature name but cannot set the requested value.

See Also:
XMLReader.getFeature(java.lang.String)

---

getFeature

```
public boolean getFeature(java.lang.String name)
                   throws SAXNotRecognizedException,
                          SAXNotSupportedException
```

Look up the value of a feature.

This will always fail if the parent is null.

Specified by:
getFeature in interface XMLReader

Parameters:
name - The feature name.

Returns:
The current value of the feature.

Throws:
SAXNotRecognizedException - If the feature value can't be assigned or retrieved from the parent.
SAXNotSupportedException - When the parent recognizes the feature name but cannot determine its value at this time.

See Also:
XMLReader.setFeature(java.lang.String, boolean)

---

setProperty

```
public void setProperty(java.lang.String name,
                        java.lang.Object value)
                 throws SAXNotRecognizedException,
```

SAXNotSupportedException

Set the value of a property.

This will always fail if the parent is null.

Specified by:
setProperty in interface XMLReader

Parameters:
name - The property name.
value - The requested property value.

Throws:
SAXNotRecognizedException - If the property value can't be assigned or retrieved from the parent.
SAXNotSupportedException - When the parent recognizes the property name but cannot set the requested value.

---

getProperty

```
public java.lang.Object getProperty(java.lang.String name)
                                throws SAXNotRecognizedException,
                                       SAXNotSupportedException
```

Look up the value of a property.

Specified by:
getProperty in interface XMLReader

Parameters:
name - The property name.

Returns:
The current value of the property.

Throws:
SAXNotRecognizedException - If the property value can't be assigned or retrieved from the parent.
SAXNotSupportedException - When the parent recognizes the property name but cannot determine its value at this time.

See Also:

XMLReader.setProperty(java.lang.String, java.lang.Object)

---

setEntityResolver

```
public void setEntityResolver(EntityResolver resolver)
```

Set the entity resolver.

Specified by:
setEntityResolver in interface XMLReader

Parameters:
resolver - The new entity resolver.

See Also:
XMLReader.getEntityResolver()

---

getEntityResolver

```
public EntityResolver getEntityResolver()
```

Get the current entity resolver.

Specified by:
getEntityResolver in interface XMLReader

Returns:
The current entity resolver, or null if none was set.

See Also:
XMLReader.setEntityResolver(org.xml.sax.EntityResolver)

---

setDTDHandler

```
public void setDTDHandler(DTDHandler handler)
```

Set the DTD event handler.

Specified by:
`setDTDHandler` in interface `XMLReader`

Parameters:
`handler` - the new DTD handler

See Also:
`XMLReader.getDTDHandler()`

---

getDTDHandler

`public DTDHandler getDTDHandler()`

Get the current DTD event handler.

Specified by:
`getDTDHandler` in interface `XMLReader`

Returns:
The current DTD handler, or null if none was set.

See Also:
`XMLReader.setDTDHandler(org.xml.sax.DTDHandler)`

---

setContentHandler

`public void setContentHandler(ContentHandler handler)`

Set the content event handler.

Specified by:
`setContentHandler` in interface `XMLReader`

Parameters:
`handler` - the new content handler

See Also:
`XMLReader.getContentHandler()`

---

getContentHandler

`public ContentHandler getContentHandler()`

Get the content event handler.

Specified by:
`getContentHandler` in interface `XMLReader`

Returns:
The current content handler, or null if none was set.

See Also:
`XMLReader.setContentHandler(org.xml.sax.ContentHandler)`

---

setErrorHandler

`public void setErrorHandler(ErrorHandler handler)`

Set the error event handler.

Specified by:
`setErrorHandler` in interface `XMLReader`

Parameters:
`handler` - the new error handler

See Also:
`XMLReader.getErrorHandler()`

---

getErrorHandler

`public ErrorHandler getErrorHandler()`

Get the current error event handler.

Specified by:
`getErrorHandler` in interface `XMLReader`

Returns:

The current error handler, or null if none was set.

See Also:
XMLReader.setErrorHandler(org.xml.sax.ErrorHandler)

---

parse

```
public void parse(InputSource input)
           throws SAXException,
                  java.io.IOException
```

Parse a document.

Specified by:
parse in interface XMLReader

Parameters:
input - The input source for the document entity.

Throws:
SAXException - Any SAX exception, possibly wrapping another exception.
java.io.IOException - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

See Also:
InputSource, XMLReader.parse(java.lang.String), XMLReader.setEntityResolver(org.xml.sax.EntityResolver), XMLReader.setDTDHandler(org.xml.sax.DTDHandler), XMLReader.setContentHandler(org.xml.sax.ContentHandler), XMLReader.setErrorHandler(org.xml.sax.ErrorHandler)

---

parse

```
public void parse(java.lang.String systemId)
           throws SAXException,
                  java.io.IOException
```

Parse a document.

Specified by:
`parse` in interface `XMLReader`

Parameters:
`systemId` - The system identifier as a fully-qualified URI.

Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
`java.io.IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

See Also:
`XMLReader.parse(org.xml.sax.InputSource)`

---

resolveEntity

```
public InputSource resolveEntity(java.lang.String publicId,
                                 java.lang.String systemId)
                          throws SAXException,
                                 java.io.IOException
```

Filter an external entity resolution.

Specified by:
`resolveEntity` in interface `EntityResolver`

Parameters:
`publicId` - The entity's public identifier, or null.
`systemId` - The entity's system identifier.

Returns:
A new InputSource or null for the default.

Throws:
`SAXException` - The client may throw an exception during processing.
`java.io.IOException` - The client may throw an I/O-related exception while obtaining the new InputSource.

See Also:
`InputSource`

---

notationDecl

```
public void notationDecl(java.lang.String name,
                         java.lang.String publicId,
                         java.lang.String systemId)
                  throws SAXException
```

Filter a notation declaration event.

Specified by:
notationDecl in interface DTDHandler

Parameters:
name - The notation name.
publicId - The notation's public identifier, or null.
systemId - The notation's system identifier, or null.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
DTDHandler.unparsedEntityDecl(java.lang.String, java.lang.String, java.lang.String, java.lang.String), Attributes

---

unparsedEntityDecl

```
public void unparsedEntityDecl(java.lang.String name,
                               java.lang.String publicId,
                               java.lang.String systemId,
                               java.lang.String notationName)
                        throws SAXException
```

Filter an unparsed entity declaration event.

Specified by:
unparsedEntityDecl in interface DTDHandler

Parameters:
name - The entity name.
publicId - The entity's public identifier, or null.
systemId - The entity's system identifier, or null.
notationName - The name of the associated notation.

Throws:

SAXException - The client may throw an exception during processing.

See Also:
DTDHandler.notationDecl(java.lang.String, java.lang.String, java.lang.String), Attributes

---

setDocumentLocator

```
public void setDocumentLocator(Locator locator)
```

Filter a new document locator event.

Specified by:
setDocumentLocator in interface ContentHandler

Parameters:
locator - The document locator.

See Also:
Locator

---

startDocument

```
public void startDocument()
                  throws SAXException
```

Filter a start document event.

Specified by:
startDocument in interface ContentHandler

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.endDocument()

---

endDocument

```
public void endDocument()
                 throws SAXException
```

Filter an end document event.

Specified by:
endDocument in interface ContentHandler

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.startDocument()

---

startPrefixMapping

```
public void startPrefixMapping(java.lang.String prefix,
                               java.lang.String uri)
                        throws SAXException
```

Filter a start Namespace prefix mapping event.

Specified by:
startPrefixMapping in interface ContentHandler

Parameters:
prefix - The Namespace prefix.
uri - The Namespace URI.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.endPrefixMapping(java.lang.String), ContentHandler.startElement(java.lang.String, java.lang.String, java.lang.String, org.xml.sax.Attributes)

---

endPrefixMapping

```
public void endPrefixMapping(java.lang.String prefix)
```

```
                    throws SAXException
```

Filter an end Namespace prefix mapping event.

Specified by:
endPrefixMapping in interface ContentHandler

Parameters:
prefix - The Namespace prefix.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.startPrefixMapping(java.lang.String, java.lang.String), ContentHandler.endElement(java.lang.String, java.lang.String, java.lang.String)

---

startElement

```
public void startElement(java.lang.String uri,
                         java.lang.String localName,
                         java.lang.String qName,
                         Attributes atts)
                  throws SAXException
```

Filter a start element event.

Specified by:
startElement in interface ContentHandler

Parameters:
uri - The element's Namespace URI, or the empty string.
localName - The element's local name, or the empty string.
qName - The element's qualified (prefixed) name, or the empty string.
atts - The element's attributes.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.endElement(java.lang.String, java.lang.String, java.lang.String), Attributes, AttributesImpl

---

endElement

```
public void endElement(java.lang.String uri,
                       java.lang.String localName,
                       java.lang.String qName)
                 throws SAXException
```

Filter an end element event.

Specified by:
`endElement` in interface `ContentHandler`

Parameters:
`uri` - The element's Namespace URI, or the empty string.
`localName` - The element's local name, or the empty string.
`qName` - The element's qualified (prefixed) name, or the empty string.

Throws:
`SAXException` - The client may throw an exception during processing.

---

characters

```
public void characters(char[] ch,
                       int start,
                       int length)
                 throws SAXException
```

Filter a character data event.

Specified by:
`characters` in interface `ContentHandler`

Parameters:
`ch` - An array of characters.
`start` - The starting position in the array.
`length` - The number of characters to use from the array.

Throws:
`SAXException` - The client may throw an exception during processing.

See Also:
ContentHandler.ignorableWhitespace(char[], int, int), Locator

---

ignorableWhitespace

```
public void ignorableWhitespace(char[] ch,
                                int start,
                                int length)
                         throws SAXException
```

Filter an ignorable whitespace event.

Specified by:
ignorableWhitespace in interface ContentHandler

Parameters:
ch - An array of characters.
start - The starting position in the array.
length - The number of characters to use from the array.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
ContentHandler.characters(char[], int, int)

---

processingInstruction

```
public void processingInstruction(java.lang.String target,
                                  java.lang.String data)
                           throws SAXException
```

Filter a processing instruction event.

Specified by:
processingInstruction in interface ContentHandler

Parameters:
target - The processing instruction target.

`data` - The text following the target.

Throws:

`SAXException` - The client may throw an exception during processing.

---

skippedEntity

```
public void skippedEntity(java.lang.String name)
                  throws SAXException
```

Filter a skipped entity event.

Specified by:

`skippedEntity` in interface `ContentHandler`

Parameters:

`name` - The name of the skipped entity.

Throws:

`SAXException` - The client may throw an exception during processing.

---

warning

```
public void warning(SAXParseException e)
             throws SAXException
```

Filter a warning event.

Specified by:

`warning` in interface `ErrorHandler`

Parameters:

`e` - The warning as an exception.

Throws:

`SAXException` - The client may throw an exception during processing.

See Also:

`SAXParseException`

---

error

```
public void error(SAXParseException e)
           throws SAXException
```

Filter an error event.

Specified by:
error in interface ErrorHandler

Parameters:
e - The error as an exception.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
SAXParseException

---

fatalError

```
public void fatalError(SAXParseException e)
                 throws SAXException
```

Filter a fatal error event.

Specified by:
fatalError in interface ErrorHandler

Parameters:
e - The error as an exception.

Throws:
SAXException - The client may throw an exception during processing.

See Also:
SAXParseException

---

11 - Dec - 08

org.xml.sax

Interface XMLReader

it has 10 methods (but 10 only need)

2 - handler setter method
2 - handler getter method
2 - setter / getter of property
2 - setter / getter of FEATURE
2 - parsing related method

All Known Subinterfaces:

XMLFilter

All Known Implementing Classes:

ParserAdapter, XMLFilterImpl

XMLReader instance can be reused once it done its parsing, even it dont have reset () method. [be careful proper close of input source, otherwise exception will be thrown]

if we want to use SAX without any validation, directly use this xmlReader from its factory.

even it dont expect any handler

---

pu

In ~~using callbacks.~~

we can use ErrorHandler, ContentHandler separately only in XMLReader. but nowhere else.
SAXParser will take only DefaultHandler

Lexical handler can be set either in XMLReader or SAXParser since it is property

~~*l documentation, is in the Public Domain, and comes*~~
~~www.saxproject.org for further information.~~

Note: despite its name, this interface does *not* extend the standard Java `Reader` interface, because reading XML is a fundamentally different activity than reading character data.

XMLReader is the interface that an XML parser's SAX2 driver must implement. This interface allows an application to set and query features and properties in the parser, to register event handlers for document processing, and to initiate a document parse.

All SAX interfaces are assumed to be synchronous: the `parse` methods must not return until parsing is complete, and readers must wait for an event-handler callback to return before reporting the next event.

~~This interface replaces the (now deprecated) SAX 1.0 `Parser` interface. The XMLReader interface contains two important enhancements over the old Parser interface (as well as some minor ones):~~

1. ~~it adds a standard way to query and set features and properties; and~~
2. ~~it adds Namespace support, which is required for many higher-level XML standards.~~

~~There are adapters available to convert a SAX1 Parser to a SAX2 XMLReader and vice-versa.~~

~~**Since:**~~

~~SAX 2.0~~

~~**Version:**~~

~~2.0.1+ (sax2r3pre1)~~

~~**Author:**~~

~~David Megginson~~

~~**See Also:**~~

~~XMLFilter, ParserAdapter, XMLReaderAdapter~~

Method Summary

ContentHandler	**getContentHandler**() Return the current content handler.
DTDHandler	~~**getDTDHandler**()~~ ~~Return the current DTD handler.~~
EntityResolver	~~**getEntityResolver**()~~ ~~Return the current entity resolver.~~
ErrorHandler	**getErrorHandler**() Return the current error handler.
boolean	**getFeature**(java.lang.String name) Look up the value of a feature flag.
java.lang.Object	**getProperty**(java.lang.String name) Look up the value of a property.
void	**parse**(InputSource input) Parse an XML document.
void	**parse**(java.lang.String systemId) Parse an XML document from a system identifier (URI).
void	**setContentHandler**(ContentHandler handler) ~~Allow an application to register a content event handler.~~
void	~~**setDTDHandler**(DTDHandler handler)~~ ~~Allow an application to register a DTD event handler.~~

what is difference between feature and property ???

Feature are Boolean value and rep the standard of XML technology

Property is Object value and specfic to Implementation

these setter methods are useful while using with SAXSource in TrAX

void	~~**setEntityResolver**(EntityResolver resolver)~~ ~~Allow an application to register an entity resolver.~~
void	**setErrorHandler**(ErrorHandler handler) Allow an application to register an error event handler.
void	**setFeature**(java.lang.String name, boolean value) Set the value of a feature flag.
void	**setProperty**(java.lang.String name, java.lang.Object value) Set the value of a property.

Method Detail

getFeature

```
boolean getFeature(java.lang.String name)
                   throws SAXNotRecognizedException,
                          SAXNotSupportedException
```

Look up the value of a feature flag.

The feature name is any fully-qualified URI. It is possible for an XMLReader to recognize a feature name but temporarily be unable to return its value. ~~Some feature values may be available only in specific contexts, such as before, during, or after a parse. Also, some feature values may not be programmatically accessible. (In the case of an adapter for SAX1 Parser, there is no implementation-independent way to expose whether the underlying parser is performing validation, expanding external entities, and so forth.)~~

All XMLReaders are required to recognize the http://xml.org/sax/features/namespaces and the http://xml.org/sax/features/namespace-prefixes feature names.

Typical usage is something like this:

```
 XMLReader r = new MySAXDriver();

                            // try to activate validation
 try {
```

```
        r.setFeature("http://xml.org/sax/features/validation", true);
    } catch (SAXException e) {
        System.err.println("Cannot activate validation.");
    }

                                // register event handlers
    r.setContentHandler(new MyContentHandler());
    r.setErrorHandler(new MyErrorHandler());

                                // parse the first document
    try {
        r.parse("http://www.foo.com/mydoc.xml");
    } catch (IOException e) {
        System.err.println("I/O exception reading XML document");
    } catch (SAXException e) {
        System.err.println("XML exception reading document.");
    }
```

~~Implementors are free (and encouraged) to invent their own features, using names built on their own URIs.~~

~~**Parameters:**~~

`name` ~~- The feature name, which is a fully-qualified URI.~~

~~**Returns:**~~

~~The current value of the feature (true or false).~~

~~**Throws:**~~

`SAXNotRecognizedException` ~~- If the feature value can't be assigned or retrieved.~~

`SAXNotSupportedException` ~~- When the XMLReader recognizes the feature name but cannot determine its value at this time.~~

~~**See Also:**~~

`setFeature(java.lang.String, boolean)`

---

setFeature

```
void setFeature(java.lang.String name,
                boolean value)
                throws SAXNotRecognizedException,
                       SAXNotSupportedException
```

Set the value of a feature flag.

~~The feature name is any fully-qualified URI. It is possible for an XMLReader to expose a feature value but to be unable to change the current value. Some feature values may be immutable or mutable only in specific contexts, such as before, during, or after a parse.~~

All XMLReaders are required to support setting http://xml.org/sax/features/namespaces to true and http://xml.org/sax/features/namespace-prefixes to false.

yes

~~Parameters:~~

`name` ~~- The feature name, which is a fully-qualified URI.~~

`value` ~~- The requested value of the feature (true or false).~~

~~Throws:~~

`SAXNotRecognizedException` ~~- If the feature value can't be assigned or retrieved.~~

`SAXNotSupportedException` ~~- When the XMLReader recognizes the feature name but cannot set the requested value.~~

~~See Also:~~

`getFeature(java.lang.String)`

---

getProperty

```
java.lang.Object getProperty(java.lang.String name)
                                 throws SAXNotRecognizedException,
                                        SAXNotSupportedException
```

~~Look up the value of a property.~~

we have to set LexicalHandler only via property

~~The property name is any fully-qualified URI. It is possible for an XMLReader to recognize a property name but temporarily be unable to return its value. Some property values may be available only in specific contexts, such as before, during, or after a parse.~~

~~XMLReaders are not required to recognize any specific property names, though an initial core set is documented for SAX2.~~

~~Implementors are free (and encouraged) to invent their own properties, using names built on their own URIs.~~

Parameters:

`name` - The property name, which is a fully-qualified URI.

~~Returns:~~

~~The current value of the property.~~

~~Throws:~~

`SAXNotRecognizedException` ~~- If the property value can't be assigned or retrieved.~~

`SAXNotSupportedException` ~~- When the XMLReader recognizes the property name but cannot determine its value at this time.~~

~~See Also:~~

~~`setProperty(java.lang.String, java.lang.Object)`~~

---

setProperty

we have to set LexicalHandler only via property

```
void setProperty(java.lang.String name,
                 java.lang.Object value)
                 throws SAXNotRecognizedException,
                        SAXNotSupportedException
```

~~Set the value of a property.~~

~~The property name is any fully-qualified URI. It is possible for an XMLReader to recognize a property name but to be unable to change the current value. Some property values may be immutable or mutable only in specific contexts, such as before, during, or after a parse.~~

~~XMLReaders are not required to recognize setting any specific property names, though a core set is defined by SAX2.~~

~~This method is also the standard mechanism for setting extended handlers.~~

~~Parameters:~~

~~`name` - The property name, which is a fully-qualified URI.~~

`value` ~~- The requested value for the property.~~

~~Throws:~~

`SAXNotRecognizedException` ~~- If the property value can't be assigned or retrieved.~~

`SAXNotSupportedException` ~~- When the XMLReader recognizes the property name but cannot set the requested value.~~

---

setEntityResolver

`void setEntityResolver(EntityResolver resolver)`

Allow an application to register an entity resolver.

If the application does not register an entity resolver, the XMLReader will perform its own default resolution.

Applications may register a new or different resolver in the middle of a parse, and the SAX parser must begin using the new resolver immediately.

Parameters:
`resolver` - The entity resolver.

See Also:
`getEntityResolver()`

---

getEntityResolver

`EntityResolver getEntityResolver()`

Return the current entity resolver.

Returns:
The current entity resolver, or null if none has been registered.

See Also:
`setEntityResolver(org.xml.sax.EntityResolver)`

---

setDTDHandler

`void setDTDHandler(DTDHandler handler)`

Allow an application to register a DTD event handler.

If the application does not register a DTD handler, all DTD events reported by the SAX parser will be silently ignored.

~~Applications may register a new or different handler in the middle of a parse, and the SAX parser must begin using the new handler immediately.~~

~~Parameters:~~

`handler` ~~- The DTD handler.~~

~~See Also:~~

`getDTDHandler()`

~~getDTDHandler~~

`DTDHandler` **~~getDTDHandler()~~**

~~Return the~~ current DTD handler.

~~Returns:~~

~~The current DTD handler, or null if none has been registered.~~

~~See Also:~~

`setDTDHandler(org.xml.sax.DTDHandler)`

---

setContentHandler

`void` **`setContentHandler`**`(ContentHandler handler)`

Allow an application to register a content event handler.

If the application does not register a content handler, all content events reported by the SAX parser will be silently ignored.

~~Applications may register a new or different handler in the middle of a parse, and the SAX parser must begin using the new handler immediately.~~

~~Parameters:~~

`handler` ~~- The content handler.~~

~~See Also:~~

`getContentHandler()`

---

getContentHandler

`ContentHandler` **`getContentHandler()`**

~~Return the current content handler.~~

~~**Returns:**~~

~~The current content handler, or null if none has been registered.~~

~~**See Also:**~~

~~`setContentHandler(org.xml.sax.ContentHandler)`~~

---

setErrorHandler

`void` **`setErrorHandler`**`(ErrorHandler handler)`

Allow an application to register an error event handler.

If the application does not register an error handler, all error events reported by the SAX parser will be silently ignored; however, normal processing may not continue. It is highly recommended that all SAX applications implement an error handler to avoid unexpected bugs.

Applications may register a new or different handler in the middle of a parse, and the SAX parser must begin using the new handler immediately.

how come ?? any sample code ??

11 - Dec - 08

Parameters:

`handler` - The error handler.

See Also:

`getErrorHandler()`

---

getErrorHandler

`ErrorHandler` **`getErrorHandler()`**

~~Return the current error handler.~~

Returns:

The current error handler, or null if none has been registered.

See Also:

`setErrorHandler(org.xml.sax.ErrorHandler)`

---

parse

```
void parse(InputSource input)
           throws java.io.IOException,
                  SAXException
```

Parse an XML document.

The application can use this method to instruct the XML reader to begin parsing an XML document from any valid input source (a character stream, a byte stream, or a URI).

Applications may not invoke this method while a parse is in progress (they should create a new XMLReader instead for each nested XML document). Once a parse is complete, an application may reuse the same XMLReader object, possibly with a different input source. Configuration of the XMLReader object (such as handler bindings and values established for feature flags and properties) is unchanged by completion of a parse, unless the definition of that aspect of the configuration explicitly specifies other behavior. (For example, feature flags or properties exposing characteristics of the document being parsed.)

During the parse, the XMLReader will provide information about the XML document through the registered event handlers.

This method is synchronous: it will not return until parsing has ended. If a client application wants to terminate parsing early, it should throw an exception.

Parameters:

`input` - The input source for the top-level of the XML document.

Throws:

`SAXException` - Any SAX exception, possibly wrapping another exception.

`java.io.IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

See Also:

`InputSource`, `parse(java.lang.String)`, `setEntityResolver(org.`

xml.sax.EntityResolver), setDTDHandler(org.xml.sax.DTDHandler), setContentHandler(org.xml.sax.ContentHandler), setErrorHandler(org.xml.sax.ErrorHandler)

---

parse

```
void parse(java.lang.String systemId)
           throws java.io.IOException,
                  SAXException
```

Parse an XML document from a system identifier (URI).

This method is a shortcut for the common case of reading a document from a system identifier. It is the exact equivalent of the following:

```
parse(new InputSource(systemId));
```

If the system identifier is a URL, it must be fully resolved by the application before it is passed to the parser.

Parameters:

systemId - The system identifier (URI).

Throws:

SAXException - Any SAX exception, possibly wrapping another exception.

java.io.IOException - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

See Also:

parse(org.xml.sax.InputSource)

---

10 - Dec - 08

2 - methods only

org.xml.sax.helpers

Class XMLReaderFactory

it has only STATIC method

```
java.lang.Object
  └org.xml.sax.helpers.XMLReaderFactory
```

so, Never create object for this XMLReaderFactory class

```
public final class XMLReaderFactory
```

extends java.lang.Object

we cannot validate with xml SCHEMA as doing in SAXParserFactory.
.
schema validation can be done as old methods like setting property or feauter

Factory for creating an XML reader.

> ~~*This module, both source code and documentation, is in the Public Domain, and comes with* ***NO WARRANTY****. See http://www.saxproject.org for further information.*~~

This class contains static methods for creating an XML reader from an explicit class name, or based on runtime defaults:

```
try {
  XMLReader myReader = XMLReaderFactory.createXMLReader();
} catch (SAXException e) {
  System.err.println(e.getMessage());
}
```

Note to Distributions bundled with parsers: You should modify the implementation of the no-arguments *createXMLReader* to handle cases where the external configuration mechanisms aren't set up. That method should do its best to return a parser when one is in the class path, even when nothing bound its class name to `org.xml.sax.driver` so those configuration mechanisms would see it.

Since:

SAX 2.0

Version:

2.0.1 (sax2r2)

Author:

David Megginson, David Brownell

All Factory method are start with "new / create" and appended by class name

newSAXParesr()
newValidator()
newValidatorHandler()
newDocumentBuilder()
newTransformer() and etc....

Method Summary

static XMLReader	**createXMLReader**() Attempt to create an XMLReader from system defaults.
static XMLReader	**createXMLReader**(java.lang.String className) Attempt to create an XML reader from a class name.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

createXMLReader

```
public static XMLReader createXMLReader()
                                throws SAXException
```

Attempt to create an XMLReader from system defaults. In environments which can support it, the name of the XMLReader class is determined by trying each these options in order, and using the first one which succeeds:

- If the system property `org.xml.sax.driver` has a value, that is used as an XMLReader class name.
- The JAR "Services API" is used to look for a class name in the *META-INF/services/org.xml.sax.driver* file in jarfiles available to the runtime.
- SAX parser distributions are strongly encouraged to provide a default XMLReader class name that will take effect only when previous options (on this list) are not successful.
- ~~Finally, if `ParserFactory.makeParser()` can return a system default SAX1 parser, that parser is wrapped in a `ParserAdapter`. (This is a migration aid for SAX1 environments, where the `org.xml.sax.parser` system property will often be usable.)~~

~~In environments such as small embedded systems, which can not support that flexibility, other mechanisms to determine the default may be used.~~

~~Note that many Java environments allow system properties to be initialized on a command line. This means that *in most cases* setting a good value for that property ensures that calls to this method will succeed, except when security policies intervene. This will also maximize application portability to older SAX environments, with less robust implementations of this method.~~

Returns:
~~A new XMLReader.~~

Throws:
`SAXException` - If no default XMLReader class can be identified and instantiated.

See Also:
`createXMLReader(java.lang.String)`

---

createXMLReader

```
public static XMLReader createXMLReader(java.lang.String className)
                                        throws SAXException
```

Attempt to create an XML reader from a class name.

Given a class name, this method attempts to load and instantiate the class as an XML reader.

Note that this method will not be usable in environments where the caller (perhaps an applet) is not permitted to load classes dynamically.

Returns:
A new XML reader.

Throws:
`SAXException` - If the class cannot be loaded, instantiated, and cast to XMLReader.

See Also:
`createXMLReader()`
