



CHAPTER 24

org.xml.sax, org.xml.sax.ext, and org.xml.sax.helpers

This chapter documents the `org.xml.sax` package and its subpackages. `org.xml.sax` defines the Simplified API for XML, or SAX, a de facto standard for parsing XML documents. The `org.xml.sax.ext` package defines optional extensions to the SAX API, and the `org.xml.sax.helpers` package defines helper classes that are often useful with SAX. These packages have been incorporated into Java 1.4 but are not defined by Sun, which is why they have an “org.xml” prefix rather than a “java” prefix.

Package `org.xml.sax`

Java 1.4

This is the core package for SAX parsing of XML documents. SAX is an “event-driven” API: a SAX parser reads an XML document and generates a stream of “SAX events” to describe the content of the document. These “events” are actually method calls made on one or more handler objects that the application has registered with the parser. The `XMLReader` interface defines the API that must be implemented by a SAX parser. `ContentHandler`, `ErrorHandler`, `EntityResolver`, and `DTDHandler` are interfaces that define handler objects. An application registers objects that implement one or more of these interfaces with the `XMLReader`.

This package defines both the SAX1 and SAX2 interfaces. The `AttributeList`, `DocumentHandler` and `Parser` interfaces, as well as the `HandlerBase` class are part of the SAX1 API and are now deprecated in favor of `Attributes`, `ContentHandler`, `XMLReader` and `org.xml.sax.helpers.DefaultHandler`.

Interfaces:

```
public interface AttributeList;  
public interface Attributes;  
public interface ContentHandler;  
public interface DocumentHandler;  
public interface DTDHandler;  
public interface EntityResolver;  
public interface ErrorHandler;
```

```

public interface Locator;
public interface Parser;
public interface XMLFilter extends XMLReader;
public interface XMLReader;

```

Classes:

```

public class HandlerBase implements DocumentHandler, DTDHandler, EntityResolver, ErrorHandler;
public class InputSource;

```

Exceptions:

```

public class SAXException extends Exception;
public class SAXNotRecognizedException extends SAXException;
public class SAXNotSupportedException extends SAXException;
public class SAXParseException extends SAXException;

```

AttributeList

Java 1.4; Deprecated in Java 1.4

org.xml.sax

This interface is part of the SAX1 API and has been deprecated in favor of the SAX2 Attributes interface, which supports XML namespaces.

```

public interface AttributeList {
    // Public Instance Methods
    public abstract int getLength();
    public abstract String getName(int i);
    public abstract String getType(String name);
    public abstract String getType(int i);
    public abstract String getValue(String name);
    public abstract String getValue(int i);
}

```

Implementations: org.xml.sax.helpers.AttributeListImpl

Passed To: DocumentHandler.startElement(), HandlerBase.startElement(),
 org.xml.sax.helpers.AttributeListImpl.{AttributeListImpl(), setAttributeList()},
 org.xml.sax.helpers.ParserAdapter.startElement()

Attributes

Java 1.4

org.xml.sax

This interface represents a list of attributes of an XML element and includes information about the attribute names, types, and values. If the SAX parser has read a DTD or schema for the document, this list of attributes will include attributes that are not explicitly specified in the document but which have a default value specified in the DTD or schema.

The most commonly used method is `getValue()`, which returns the value of a named attribute (there is also a version of this method that returns the value of a numbered attribute; it is discussed later). If the SAX parser is not processing namespaces, you can use the one-argument version of `getValue()`. Otherwise, use the two argument version to specify the URI that uniquely identifies the namespace, and the “local name” of the desired attribute within that namespace. The `getType()` methods are similar, except that they return the type of the named attribute, rather than its value. Note that `getType()` can only return useful information if the parser has read a DTD or schema for the document and knows the type of each attribute.

Attributes

In XML documents the attributes of a tag can appear in any order. `Attributes` objects make no attempt to preserve the document source order of the tags. Nevertheless, it does impose an ordering on the attributes so that you can loop through them. `getLength()` returns the number of elements in the list. There are versions of `getValue()` and `getType()` that return the value and type of the attribute at a specified position in the list. You can also query the name of the attribute at a specified position, although the way you do this depends on whether the parser handles namespaces or not. If it does not process namespaces, use `getQName()` to get the name at a specified position. Otherwise, use `getURI()` and `getLocalName()` to obtain the URI and local name pair for the numbered attribute. Note that `getQName()` may return the empty string when namespace processing is on, and `getLocalName()` may return the empty string if namespace processing is off.

```
public interface Attributes {  
    // Public Instance Methods  
    public abstract int getIndex(String qName);  
    public abstract int getIndex(String uri, String localPart);  
    public abstract int getLength();  
    public abstract String getLocalName(int index);  
    public abstract String getQName(int index);  
    public abstract String getType(String qName);  
    public abstract String getType(int index);  
    public abstract String getType(String uri, String localName);  
    public abstract String getURI(int index);  
    public abstract String getValue(String qName);  
    public abstract String getValue(int index);  
    public abstract String getValue(String uri, String localName);  
}
```

Implementations: `org.xml.sax.helpers.AttributesImpl`

Passed To: `org.xml.sax.ContentHandler.startElement()`,
`org.xml.sax.helpers.AttributesImpl.{AttributesImpl(), setAttributes()}`,
`org.xml.sax.helpers.DefaultHandler.startElement()`, `org.xml.sax.helpers.XMLFilterImpl.startElement()`,
`org.xml.sax.helpers.XMLReaderAdapter.startElement()`

ContentHandler

Java 1.4

`org.xml.sax`

This interface is the key one for XML parsing with the SAX API. An `XMLReader` tells your application about the content of the XML document it is parsing by invoking the various methods of the `ContentHandler` interface. In order to parse documents with SAX, you must implement this interface to define methods that take whatever actions are necessary when they are invoked by the parser. Because this interface is so critical to the SAX API, the methods are explained individually:

`setDocumentLocator()`

The parser usually calls this method (but is not required to do so) before calling any others to pass a `Locator` object to the `ContentHandler`. `Locator` defines methods that return the current line and column number of the document being parsed, and if the parser supplies a `Locator` object, it guarantees that its methods will return valid values during any other `ContentHandler` invocations that follow. A `ContentHandler` can call the methods of this object when printing error messages, for example.

`startDocument()`, `endDocument()`

The parser calls these methods once, at the beginning and end of parsing. `startDocument()` is the first method called except for the optional `setDocumentLocator()` call, and `endDocument()` is always the last method call on a `ContentHandler`.

startElement(), endElement()

The parser calls these methods for each start tag and end tag it encounters. Both are passed three arguments describing the name of the tag: if the parser is doing namespace processing, then the first two arguments of both methods return the URI that uniquely identifies the namespace, and the local name of the tag within that namespace. If the parser is not doing namespace parsing, then the third argument provides the full name of the tag. In addition to these tag name arguments, **startElement()** is also passed an **Attributes** object that describes the attributes of the tag.

characters()

This method is invoked to tell the application that the parser has found a string of text in the XML document. The text is contained within the specified character array, at the specified start position, and continuing for the specified number of characters.

ignorableWhitespace()

This method is like **characters()**, but parsers may use it to tell the application about “ignorable whitespace” in XML element content.

processingInstruction()

The parser calls this method to tell the application that it has encountered an XML Processing Instruction (or PI) with the specified target and data strings.

skippedEntity()

If the XML parser does encounters an entity in the document, but does not expand and parse its content, then it tells the application about it by passing the name of the entity to this method.

startPrefixMapping(), endPrefixMapping()

These methods to tell the application about a namespace mapping from the specified prefix to the specified namespace URI.

DTDHandler is another interface like **ContentHandler**. An application can implement this interface to receive notification of DTD-related events from the parser. Similarly, the **org.xml.sax.ext** package defines two “extension” interfaces that can be used (if the parser supports these extensions) to obtain even more information about the document (such as comments and CDATA sections) and about the DTD (including the full set of element, attribute and entity declarations). The **org.xml.sax.helpers.DefaultHandler** class is a useful one. It implements **ContentHandler** and three other interfaces that are commonly used with the **XMLReader** class and provides empty implementations of all their methods. Applications can subclass **DefaultHandler** only need to override the methods they care about. This is usually more convenient than implementing the interfaces directly.

```
public interface ContentHandler {
    // Public Instance Methods
    public abstract void characters(char[] ch, int start, int length) throws SAXException;
    public abstract void endDocument() throws SAXException;
    public abstract void endElement(String namespaceURI, String localName, String qName) throws SAXException;
    public abstract void endPrefixMapping(String prefix) throws SAXException;
    public abstract void ignorableWhitespace(char[] ch, int start, int length) throws SAXException;
    public abstract void processingInstruction(String target, String data) throws SAXException;
    public abstract void setDocumentLocator(Locator locator);
    public abstract void skippedEntity(String name) throws SAXException;
    public abstract void startDocument() throws SAXException;
    public abstract void startElement(String namespaceURI, String localName, String qName,
        org.xml.sax.Attributes atts) throws SAXException;
```

ContentHandler

```
public abstract void startPrefixMapping(String prefix, String uri) throws SAXException;  
}
```

Implementations: javax.xml.transform.sax.TemplatesHandler,
javax.xml.transform.sax.TransformerHandler, org.xml.sax.helpers.DefaultHandler,
org.xml.sax.helpers.XMLFilterImpl, org.xml.sax.helpers.XMLReaderAdapter

Passed To: javax.xml.transform.sax.SAXResult.{SAXResult(), setHandler()},
XMLReader.setContentHandler(), org.xml.sax.helpers.ParserAdapter.setContentHandler(),
org.xml.sax.helpers.XMLFilterImpl.setContentHandler()

Returned By: javax.xml.transform.sax.SAXResult.getHandler(), XMLReader.getContentHandler(),
org.xml.sax.helpers.ParserAdapter.getContentHandler(),
org.xml.sax.helpers.XMLFilterImpl.getContentHandler()

DocumentHandler

Java 1.4; Deprecated in Java 1.4

org.xml.sax

This interface is part of the SAX1 API and has been deprecated in favor of the SAX2 ContentHandler interface, which supports XML namespaces.

```
public interface DocumentHandler {  
    // Public Instance Methods  
    public abstract void characters(char[] ch, int start, int length) throws SAXException;  
    public abstract void endDocument() throws SAXException;  
    public abstract void endElement(String name) throws SAXException;  
    public abstract void ignorableWhitespace(char[] ch, int start, int length) throws SAXException;  
    public abstract void processingInstruction(String target, String data) throws SAXException;  
    public abstract void setDocumentLocator(Locator locator);  
    public abstract void startDocument() throws SAXException;  
    public abstract void startElement(String name, org.xml.sax.AttributeList atts) throws SAXException;  
}
```

Implementations: HandlerBase, org.xml.sax.helpers.ParserAdapter

Passed To: org.xml.sax.Parser.setDocumentHandler(),
org.xml.sax.helpers.XMLReaderAdapter.setDocumentHandler()

DTDHandler

Java 1.4

org.xml.sax

This interface defines methods that an application can implement in order to receive notification from a XMLReader about notation and unparsed entity declarations in the DTD of an XML document. Notations and unparsed entities are two of the most obscure features of XML, and they (and this interface) are not frequently used. To use a DTDHandler, define a class that implements the interface, (or simply subclass the helper class org.xml.sax.helpers.DefaultHandler) and pass an instance of that class to the setDTDHandler() method of an XMLReader. Then, if the parser encounters any notation or unparsed entity declarations in the DTD of the document, it will invoke the notationDecl() or unparsedEntityDecl() method that you have supplied. Unparsed entities can appear later in a document as the value of an attribute, so if your application cares about them, it should somehow make a note of the entity name and system ID for later use.

```
public interface DTDHandler {  
    // Public Instance Methods  
    public abstract void notationDecl(String name, String publicId, String systemId) throws SAXException;
```

```
public abstract void unparsedEntityDecl(String name, String publicId, String systemId, String notationName)
    throws SAXException;
}
```

Implementations: javax.xml.transform.sax.TransformerHandler, HandlerBase, org.xml.sax.helpers.DefaultHandler, org.xml.sax.helpers.XMLFilterImpl

Passed To: org.xml.sax.Parser.setDTDHandler(), XMLReader.setDTDHandler(), org.xml.sax.helpers.ParserAdapter.setDTDHandler(), org.xml.sax.helpers.XMLFilterImpl.setDTDHandler(), org.xml.sax.helpers.XMLReaderAdapter.setDTDHandler()

Returned By: XMLReader.getDTDHandler(), org.xml.sax.helpers.ParserAdapter.getDTDHandler(), org.xml.sax.helpers.XMLFilterImpl.getDTDHandler()

EntityResolver

Java 1.4

org.xml.sax

An application can implement this interface to help the parser resolve external entities, if required. If you pass an `EntityResolver` instance to the `setEntityResolver()` method of an `XMLReader`, then the parser will call the `resolveEntity()` method whenever it needs to read an external entity. This method should use the public identifier or system identifier to return an `InputSource` that the parser can use to read the content of the external entity. If the external entity includes a valid system identifier, then the parser can read it directly without the need for an `EntityResolver`, but this interface is still useful for mapping network URLs to locally cached copies, or for mapping public identifiers to local files, for example. The helper class `org.xml.sax.helpers.DefaultHandler` includes a stub implementation of this interface, so if you subclass `DefaultHandler` you can override its `resolveEntity()` method.

```
public interface EntityResolver {
    // Public Instance Methods
    public abstract InputSource resolveEntity(String publicId, String systemId) throws SAXException,
        java.io.IOException;
}
```

Implementations: HandlerBase, org.xml.sax.helpers.DefaultHandler, org.xml.sax.helpers.XMLFilterImpl

Passed To: javax.xml.parsers.DocumentBuilder.setEntityResolver(), org.xml.sax.Parser.setEntityResolver(), XMLReader.setEntityResolver(), org.xml.sax.helpers.ParserAdapter.setEntityResolver(), org.xml.sax.helpers.XMLFilterImpl.setEntityResolver(), org.xml.sax.helpers.XMLReaderAdapter.setEntityResolver()

Returned By: XMLReader.getEntityResolver(), org.xml.sax.helpers.ParserAdapter.getEntityResolver(), org.xml.sax.helpers.XMLFilterImpl.getEntityResolver()

ErrorHandler

Java 1.4

org.xml.sax

Before parsing an XML document, an application should provide an implementation of this interface to the `XMLReader` by calling the `setErrorHandler()` method of the `XMLReader`. If the reader needs to issue a warning or report an error or fatal error, it will call the appropriate method of the `ErrorHandler` object you supplied. The `error()` method is used to report recoverable errors, such as document validity problems. The parser continues parsing after calling `error()`. The `fatalError()` method is used to report nonrecoverable errors, such as well-formedness problems. The parser may not continue parsing after

ErrorHandler

calling `fatalError()`. An `ErrorHandler` object may respond to warnings, errors, and fatal errors however it likes, and may throw exceptions from these methods.

Instead of implementing this interface directly, you may also subclass the helper class `org.xml.sax.helpers.DefaultHandler` and override the error reporting methods it provides. The `warning()` and `error()` methods of a `DefaultHandler` do nothing, and the `fatalError()` method throws the `SAXParseException` object that was passed to it.

```
public interface ErrorHandler {  
    // Public Instance Methods  
    public abstract void error(SAXParseException exception) throws SAXException;  
    public abstract void fatalError(SAXParseException exception) throws SAXException;  
    public abstract void warning(SAXParseException exception) throws SAXException;  
}
```

Implementations: `HandlerBase`, `org.xml.sax.helpers.DefaultHandler`,
`org.xml.sax.helpers.XMLFilterImpl`

Passed To: `javax.xml.parsers.DocumentBuilder.setErrorHandler()`,
`org.xml.sax.Parser.setErrorHandler()`, `XMLReader.setErrorHandler()`,
`org.xml.sax.helpers.ParserAdapter.setErrorHandler()`,
`org.xml.sax.helpers.XMLFilterImpl.setErrorHandler()`,
`org.xml.sax.helpers.XMLReaderAdapter.setErrorHandler()`

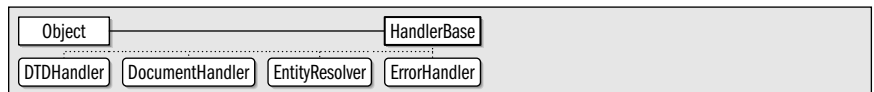
Returned By: `XMLReader.getErrorHandler()`, `org.xml.sax.helpers.ParserAdapter.getErrorHandler()`,
`org.xml.sax.helpers.XMLFilterImpl.getErrorHandler()`

HandlerBase

Java 1.4; Deprecated in Java 1.4

`org.xml.sax`

This class is part of the SAX1 API and has been deprecated in favor of the SAX2 `org.xml.sax.helpers.DefaultHandler` class.



```
public class HandlerBase implements DocumentHandler, DTDHandler, EntityResolver, ErrorHandler {  
    // Public Constructors  
    public HandlerBase();  
    // Methods Implementing DocumentHandler  
    public void characters(char[] ch, int start, int length) throws SAXException; empty  
    public void endDocument() throws SAXException; empty  
    public void endElement(String name) throws SAXException; empty  
    public void ignorableWhitespace(char[] ch, int start, int length) throws SAXException; empty  
    public void processingInstruction(String target, String data) throws SAXException; empty  
    public void setDocumentLocator(Locator locator); empty  
    public void startDocument() throws SAXException; empty  
    public void startElement(String name, org.xml.sax.AttributeList attributes) throws SAXException; empty  
    // Methods Implementing DTDHandler  
    public void notationDecl(String name, String publicId, String systemId); empty  
    public void unparsedEntityDecl(String name, String publicId, String systemId, String notationName); empty  
    // Methods Implementing EntityResolver  
    public InputSource resolveEntity(String publicId, String systemId) throws SAXException; constant  
    // Methods Implementing ErrorHandler  
    public void error(SAXParseException e) throws SAXException; empty  
    public void fatalError(SAXParseException e) throws SAXException;  
    public void warning(SAXParseException e) throws SAXException; empty  
}
```

Passed To: javax.xml.parsers.SAXParser.parse()

InputSource

Java 1.4

org.xml.sax

This simple class describes a source of input for an `XMLReader`. An `InputSource` object can be passed to the `parse()` method of `XMLReader`, and is also the return value of the `EntityResolver.resolveEntity()` method.

Create an `InputSource()` with one of the constructor methods, specifying the system identifier (a URL) of the file to be parsed, or specifying a byte or character stream that the parser should read the document from. In addition to calling the constructor, you may also want to call `setSystemId()` to specify and/or `setPublicId()` to provide identifiers for the document being parsed. Having a filename or URL is useful if an error arises, and your `ErrorHandler` object needs to print an error message, for example. If you specify the document to parse as a URL or as a byte stream, you can also call `setEncoding()` to specify the character encoding of the document. The parser will use this encoding value if you supply it, but XML documents are supposed to describe their own encoding in the `<?xml?>` declaration, so the parser ought to be able to determine the encoding of the document even if you do not call `setEncoding()`.

This class allows you to specify more than one input source. The `XMLReader` will first call `getCharacterStream()` and use the returned `Reader` if there is one. If that method returns `false`, then it calls `getByteStream()` and uses the `InputStream` it returns. Finally, if no character or byte stream is found, then the parser will call `getSystemId()` and will attempt to read an XML document from the returned URL.

An `XMLReader` will never use any of the `set()` methods to modify the state of an `InputSource` object.

```
public class InputSource {
    // Public Constructors
    public InputSource();
    public InputSource(java.io.Reader characterStream);
    public InputSource(java.io.InputStream byteStream);
    public InputSource(String systemId);
    // Property Accessor Methods (by property name)
    public java.io.InputStream getByteStream();           default:null
    public void setByteStream(java.io.InputStream byteStream);
    public java.io.Reader getCharacterStream();           default:null
    public void setCharacterStream(java.io.Reader characterStream);
    public String getEncoding();                         default:null
    public void setEncoding(String encoding);
    public String getPublicId();                         default:null
    public void setPublicId(String publicId);
    public String getSystemId();                         default:null
    public void setSystemId(String systemId);
}
```

Passed To: javax.xml.parsers.DocumentBuilder.parse(), javax.xml.parsers.SAXParser.parse(), javax.xml.transform.sax.SAXSource.{SAXSource(), setInputSource()}, org.xml.sax.Parser.parse(), XMLReader.parse(), org.xml.sax.helpers.ParserAdapter.parse(), org.xml.sax.helpers.XMLFilterImpl.parse(), org.xml.sax.helpers.XMLReaderAdapter.parse()

Returned By: javax.xml.transform.sax.SAXSource.{getInputSource(), sourceToInputSource()}, EntityResolver.resolveEntity(), HandlerBase.resolveEntity(), org.xml.sax.helpers.DefaultHandler.resolveEntity(), org.xml.sax.helpers.XMLFilterImpl.resolveEntity()

Locator

Java 1.4

org.xml.sax

An `XMLReader` may pass an object that implements this interface to the application by calling the `setDocumentLocator()` method of the application's `ContentHandler` object before it invokes any other methods of that `ContentHandler`. The `ContentHandler` can use methods of this `Locator` object from within any of the other methods called by the parser in order to determine what document the parser is parsing and what line number and column number it is parsing at. This information is particularly useful when displaying error or warning messages, for example. `getSystemId()` and `getPublicId()` return the system and public identifiers of the document being parsed, if this information is available to the parser, and otherwise return null. `getLineNumber()` and `getColumnNumber()` return the line number and column number of the next character that the parser will read (line and column numbers are numbered starting at 1, not at 0). The parser is allowed to return an approximate value from these methods, or to return -1 if it does not track line and column numbers.

```
public interface Locator {
    // Public Instance Methods
    public abstract int getColumnNumber();
    public abstract int getLineNumber();
    public abstract String getPublicId();
    public abstract String getSystemId();
}
```

Implementations: org.xml.sax.helpers.LocatorImpl

Passed To: org.xml.sax.ContentHandler.setDocumentLocator(),
 DocumentHandler.setDocumentLocator(), HandlerBase.setDocumentLocator(),
 SAXParseException.SAXParseException(), org.xml.sax.helpers.DefaultHandler.setDocumentLocator(),
 org.xml.sax.helpers.LocatorImpl.LocatorImpl(), org.xml.sax.helpers.ParserAdapter.setDocumentLocator(),
 org.xml.sax.helpers.XMLFilterImpl.setDocumentLocator(),
 org.xml.sax.helpers.XMLReaderAdapter.setDocumentLocator()

Parser

Java 1.4; Deprecated in Java 1.4

org.xml.sax

This interface is part of the SAX1 API and has been deprecated in favor of the SAX2 `XMLReader` interface, which supports XML namespaces.

```
public interface Parser {
    // Public Instance Methods
    public abstract void parse(InputSource source) throws SAXException, java.io.IOException;
    public abstract void parse(String systemId) throws SAXException, java.io.IOException;
    public abstract void setDocumentHandler(DocumentHandler handler);
    public abstract void setDTDHandler(DTDHandler handler);
    public abstract void setEntityResolver(EntityResolver resolver);
    public abstract void setErrorHandler(ErrorHandler handler);
    public abstract void setLocale(java.util.Locale locale) throws SAXException;
}
```

Implementations: org.xml.sax.helpers.XMLReaderAdapter

Passed To: org.xml.sax.helpers.ParserAdapter.ParserAdapter()

Returned By: javax.xml.parsers.SAXParser.getParser(),
 org.xml.sax.helpers.ParserFactory.makeParser()

SAXException

Java 1.4

org.xml.sax

serializable checked

This signals a problem while parsing an XML document. This class serves as the general superclass for more specific types of SAX exceptions. The `parse()` method of an XML-Reader can throw an exception of this type. The application can also throw a SAXException from any of the handler methods (of `ContentHandler` and `ErrorHandler`, for example) invoked by the parser.



```

public class SAXException extends Exception {
// Public Constructors
    public SAXException(String message);
    public SAXException(Exception e);
    public SAXException(String message, Exception e);
// Public Instance Methods
    public Exception getException();
// Public Methods Overriding Throwable
    public String getMessage();
    public String toString();
}
  
```

Subclasses: SAXNotRecognizedException, SAXNotSupportedException, SAXParseException

Thrown By: Too many methods to list.

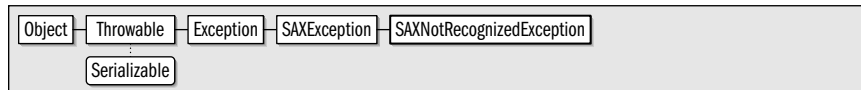
SAXNotRecognizedException

Java 1.4

org.xml.sax

serializable checked

This signals that the parser does not recognize a feature or property name. See the `setFeature()` and `setProperty()` methods of `XMLReader`.



```

public class SAXNotRecognizedException extends SAXException {
// Public Constructors
    public SAXNotRecognizedException(String message);
}
  
```

Thrown By: Too many methods to list.

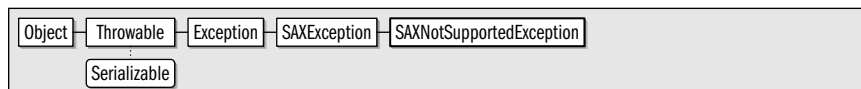
SAXNotSupportedException

Java 1.4

org.xml.sax

serializable checked

This signals that the parser recognizes, but does not support, a named feature or property. The property or feature may be entirely unsupported, or it may be read-only, in which case this exception will be thrown by the `setFeature()` or `setProperty()` method, but not by the corresponding `getFeature()` or `getProperty()` method of `XMLReader`.



SAXNotSupportedException

```
public class SAXNotSupportedException extends SAXException {  
    // Public Constructors  
    public SAXNotSupportedException(String message);  
}
```

Thrown By: Too many methods to list.

SAXParseException

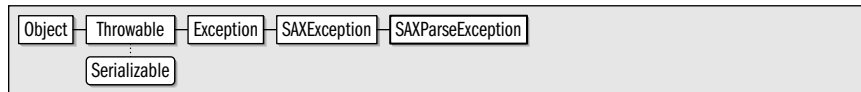
Java 1.4

org.xml.sax

serializable checked

An exception of this type signals an XML parsing error or warning. **SAXParseException** includes methods to return the system and public identifiers of the document in which the error or warning occurred, as well as methods to return the approximate line number and column number at which it occurred. A parser is not required to obtain or track all of this information, and the methods may return null or -1 if the information is not available. (See **Locator** for more information.)

Exceptions of this type are usually thrown by the application from the methods of the **ErrorHandler** interface. The parser never throws a **SAXParseException** itself, but does pass an appropriately initialized instance of this class to each of the **ErrorHandler** methods. It is up to the application's **ErrorHandler** object to decide whether to actually throw the exception, however.



```
public class SAXParseException extends SAXException {  
    // Public Constructors  
    public SAXParseException(String message, Locator locator);  
    public SAXParseException(String message, Locator locator, Exception e);  
    public SAXParseException(String message, String publicId, String systemId, int lineNumber, int columnNumber);  
    public SAXParseException(String message, String publicId, String systemId, int lineNumber, int columnNumber,  
        Exception e);  
  
    // Public Instance Methods  
    public int getColumnNumber();  
    public int getLineNumber();  
    public String getPublicId();  
    public String getSystemId();  
}
```

Passed To: **ErrorHandler**.{**error()**, **fatalError()**, **warning()**}, **HandlerBase**.{**error()**, **fatalError()**, **warning()**}, **org.xml.sax.helpers.DefaultHandler**.{**error()**, **fatalError()**, **warning()**}, **org.xml.sax.helpers.XMLFilterImpl**.{**error()**, **fatalError()**, **warning()**}

XMLFilter

Java 1.4

org.xml.sax

An **XMLFilter** extends **XMLReader** and behaves like an **XMLReader** except that instead of parsing a document itself, it filters the SAX events provided by a “parent” **XMLReader** object. Use the **setParent()** method to link an **XMLFilter** object to the **XMLReader** that it is to serve as a filter for.

An **XMLFilter** serves as both a source of SAX events, and also as a recipient of those events, so an implementation must implement **ContentHandler** and related interfaces so that it can obtain events from the parent object, filter them, and then pass the filtered events on to the **ContentHandler** object that was registered on the filter. See the helper class **org.xml.sax.helpers.XMLFilterImpl** for a bare-bones implementation of an **XMLFilter** that

implements the `XMLReader` interface and the `ContentHandler` and related handler interfaces. `XMLFilterImpl` does no filtering—it simply passes passes all of its method invocations through. You can subclass it and override only the methods that need filtering.

```

XMLReader XMLFilter
public interface XMLFilter extends XMLReader {
    // Public Instance Methods
    public abstract XMLReader getParent();
    public abstract void setParent(XMLReader parent);
}

Implementations: org.xml.sax.helpers.XMLFilterImpl
Returned By: javax.xml.transform.sax.SAXTransformerFactory.newXMLFilter()

```

XMLReader

Java 1.4

`org.xml.sax`

This interface defines the methods that must be implemented by a SAX2 XML parser. Since it is an interface, `XMLReader` cannot define a constructor for creating an `XMLReader`. To obtain an `XMLReader` object, you can instantiate some implementation-specific class that implements this interface. Alternatively, you can keep your code independent of any specific parser implementation by using the `SAXParserFactory` and `SAXParser` classes of the `javax.xml.parsers` package. See those classes for more details. Note that the `XMLReader` interface has no relationship to the `java.io.Reader` class or any other character stream classes.

Once you have obtained an `XMLReader` instance, you must register handler objects on it, so that it can invoke methods on those handlers to notify your application of the results of its parsing. All applications should register a `ContentHandler` and an `ErrorHandler` with `setContentHandler()` and `setErrorHandler()`. Some applications may also want to register an `EntityResolver` and/or a `DTDHandler`. Applications can also register `DeclHandler` and `LexicalHandler` objects from the `org.xml.sax.ext` package, if the parser implementation supports these extension handler interfaces. `DeclHandler` and `LexicalHandler` objects are registered with `setProperty()`, as explained below.

In addition to registering handler objects for an `XMLReader`, you may also want to configure the behavior of the parser using `setFeature()` and `setProperty()`. Features and properties are both name/value pairs. For uniqueness, the names of features and properties are expressed as URLs (the URLs usually do not have any web content associated with them: they are merely unique identifiers). Features have boolean values, and properties have arbitrary object values. Features and properties are an extension mechanism, allowing an application to specify implementation-specific details about how the parser should behave. But there are also several “standard” features and properties that are supported by many (or all) SAX parsers. They are listed below. If a parser does not recognize the name of a feature or property, the `setFeature()` and `setProperty()` methods (as well as the corresponding `getFeature()` and `getProperty()` query methods) throw a `SAXNotRecognizedException`. If the parser recognizes the name of a feature or property, but does not support the feature or property, the methods instead throw a `SAXNotSupportedException`. This exception is also thrown by the `set` methods when the parser allows the feature or property to be queried but not set.

The standard features are the following. Their names are all URLs that begin with the prefix “`http://www.xml.org/sax/features/`”. For brevity, this prefix has been omitted

XMLReader

below. Note that only two of these features must be supported by all parsers. The others may or may not be supported in any given implementation:

namespaces

If **true** (the default), the parser supports namespaces and provides the namespace URI and localname for element and attribute names. Support for this feature is required in all parser implementations.

namespace-prefixes

If **true**, the parser provides the qualified name (or “qName”) that for element and attribute names. A qName consists of a namespace prefix, a colon, and the local name. The default value of this feature is **false**, and support for the feature is required in all parser implementations.

validation

If **true**, the parser will validate XML documents, and will read all external entities.

external-general-entities

If **true**, the parser handles external general entities. This is always **true** if the validation feature is **true**.

external-parameter-entities

If **true**, the parser handles external parameter entities. This is always **true** if the validation feature is **true**.

lexical-handler/parameter-entities

If **true**, the parser will report the beginning and end of parameter entities to the `LexicalHandler` extension interface.

string-interning

If **true**, the parser will use the `String.intern()` method for all strings (element, attribute, entity and notation names, and namespace prefixes and URIs) it returns. If the application does the same, it can use `==` equality testing for these strings rather than using the more expensive `equals()` method.

The standard properties are the following. Like the features, their names are all URLs that begin with the prefix “`http://www.xml.org/sax/properties/`” (omitted in the following list). Note that support for all of these properties is optional.

declaration-handler

An `org.xml.sax.ext.DeclHandler` object to which the parser will report the contents of the DTD.

lexical-handler

An `org.xml.sax.ext.LexicalHandler` object on which the parser will make method calls to describe the lexical structure (such as comments and CDATA sections) of the XML document.

xml-string

This is a read-only property that can be queried only from within a handler method invoked by the parser. The value of this property is a `String` that contains the document content that triggered the current handler invocation.

dom-node

An `XMLReader` that “parses” a DOM tree rather than the textual form of an XML document uses the value of this property as the `org.w3c.dom.Node` object at which it should begin parsing.

Finally, after you have obtained an `XMLReader` object, have queried and configured its features and properties, and have set a `ContentHandler`, `ErrorHandler`, and any other required handler objects, you are ready to parse an XML document. Do this by calling one of the `parse()` methods, specifying the document to parse either as a system identifier (a URL) or as an `InputStream` object (which also allows the use of streams).

```
public interface XMLReader {
    // Public Instance Methods
    public abstract org.xml.sax.ContentHandler getContentHandler();
    public abstract DTDHandler getDTDHandler();
    public abstract EntityResolver getEntityResolver();
    public abstract ErrorHandler getErrorHandler();
    public abstract boolean getFeature(String name) throws SAXNotRecognizedException, SAXNotSupportedException;
    public abstract Object getProperty(String name) throws SAXNotRecognizedException, SAXNotSupportedException;
    public abstract void parse(String systemId) throws java.io.IOException, SAXException;
    public abstract void parse(InputStream input) throws java.io.IOException, SAXException;
    public abstract void setContentHandler(org.xml.sax.ContentHandler handler);
    public abstract void setDTDHandler(DTDHandler handler);
    public abstract void setEntityResolver(EntityResolver resolver);
    public abstract void setErrorHandler(ErrorHandler handler);
    public abstract void setFeature(String name, boolean value) throws SAXNotRecognizedException, SAXNotSupportedException;
    public abstract void setProperty(String name, Object value) throws SAXNotRecognizedException, SAXNotSupportedException;
}
```

Implementations: `XMLFilter`, `org.xml.sax.helpers.ParserAdapter`

Passed To: `javax.xml.transform.sax.SAXSource`.`{SAXSource(), setXMLReader(), XMLFilter.setParent(), org.xml.sax.helpers.XMLFilterImpl.{setParent(), XMLFilterImpl(), org.xml.sax.helpers.XMLReaderAdapter.XMLReaderAdapter()}}`

Returned By: `javax.xml.parsers.SAXParser.getXMLReader(), javax.xml.transform.sax.SAXSource.getXMLReader(), XMLFilter.getParent(), org.xml.sax.helpers.XMLFilterImpl.getParent(), org.xml.sax.helpers.XMLReaderFactory.createXMLReader()`

Package org.xml.sax.ext

Java 1.4

This package defines extensions to the basic SAX2 API. Neither SAX parsers nor SAX applications are required to support these extensions, but when they do, the interfaces defined here provide a standard way for the parser to provide additional information about an XML document to the application. `DeclHandler` defines methods for reporting the content of a DTD, and `LexicalHandler` defines methods for reporting the lexical structure of an XML document.

At the time of this writing, a “SAX2 Extensions 1.1” release is in preparation. The current beta version adds three new interfaces to this package: `Attributes2`, `EntityResolver2`, and `Locator2`. Because these new extensions are not yet released, they are not documented here. See <http://www.saxproject.org> for details.

Interfaces:

```
public interface DeclHandler;
public interface LexicalHandler;
```

DeclHandler

Java 1.4

org.xml.sax.ext

This extension interface defines methods that a SAX parser can call to notify an application about element, attribute, and entity declarations in a DTD. If your application requires this information about a DTD, then pass an object that implements this interface to the `setProperty()` method of an `XMLReader`, using the property name “`http://www.xml.org/sax/properties/declaration-handler`”. Because this is an extension handler, SAX parsers are not required to support it, and may throw a `SAXNotRecognizedException` or a `SAXNotSupportedException` when you attempt to register a `DeclHandler`.

```
public interface DeclHandler {
    // Public Instance Methods
    public abstract void attributeDecl(String eName, String aName, String type, String valueDefault, String value)
        throws org.xml.sax.SAXException;
    public abstract void elementDecl(String name, String model) throws org.xml.sax.SAXException;
    public abstract void externalEntityDecl(String name, String publicId, String systemId)
        throws org.xml.sax.SAXException;
    public abstract void internalEntityDecl(String name, String value) throws org.xml.sax.SAXException;
}
```

LexicalHandler

Java 1.4

org.xml.sax.ext

This extension interface defines methods that a SAX parser can call to notify an application about the lexical structure of an XML document. If your application requires this kind of information (for example if it wants to create a new document that has a similar structure to the one it reads), then pass an object that implements this interface to the `setProperty()` method of an `XMLReader`, using the property name “`http://www.xml.org/sax/properties/lexical-handler`”. Because this is an extension handler, SAX parsers are not required to support it, and may throw a `SAXNotRecognizedException` or a `SAXNotSupportedException` when you attempt to register a `DeclHandler`.

If a `LexicalHandler` is successfully registered on an `XMLReader`, then the parser will call `startDTD()` and `endDTD()` to report the beginning and end of the document's DTD. It will call `startCDATA()` and `endCDATA()` to report the start and end of a `CDATA` section. The content of the `CDATA` section will be reported through the `characters()` method of the `ContentHandler` interface. When the parser expands an entity, it first calls `startEntity()` to specify the name of the entity it is about to expand, and then calls `endEntity()` when the entity expansion is complete. Finally, whenever the parser encounters an XML comment, it calls the `comment()` method.

```
public interface LexicalHandler {
    // Public Instance Methods
    public abstract void comment(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public abstract void endCDATA() throws org.xml.sax.SAXException;
    public abstract void endDTD() throws org.xml.sax.SAXException;
    public abstract void endEntity(String name) throws org.xml.sax.SAXException;
    public abstract void startCDATA() throws org.xml.sax.SAXException;
    public abstract void startDTD(String name, String publicId, String systemId) throws org.xml.sax.SAXException;
    public abstract void startEntity(String name) throws org.xml.sax.SAXException;
}
```

Implementations: javax.xml.transform.sax.TransformerHandler

Passed To: javax.xml.transform.sax.SAXResult.setLexicalHandler()

Returned By: javax.xml.transform.sax.SAXResult.getLexicalHandler()

Package org.xml.sax.helpers

Java 1.4

This package contains utility classes that are useful for programmers working with SAX parsers. `DefaultHandler` is the most commonly used: it is a default implementation of the four standard handler interfaces, suitable for easy subclassing by an application. `XMLReaderFactory` provides a layer implementation-independence, allowing an application to use an `XMLReader` implementation specified in a system property. `XMLFilterImpl` is a no-op implementation of the `XMLFilter` interface that also implements the various handler interfaces necessary to connect the filter to its “parent” `XMLReader`. It does no filtering of its own, but is easy to subclass to add filtering. If you need to work with legacy APIs that expect or return SAX1 Parser objects, you can use `ParserAdapter` to make a Parser object behave like a SAX2 `XMLReader` object, or use an `XMLReaderAdapter` to make an `XMLReader` behave like a Parser.

Classes:

```
public class AttributeListImpl implements org.xml.sax.AttributeList;
public class AttributesImpl implements org.xml.sax.Attributes;
public class DefaultHandler implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler,
    org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler;
public class LocatorImpl implements org.xml.sax.Locator;
public class NamespaceSupport;
public class ParserAdapter implements org.xml.sax.DocumentHandler, org.xml.sax.XMLReader;
public class ParserFactory;
public class XMLFilterImpl implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler,
    org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, org.xml.sax.XMLFilter;
public class XMLReaderAdapter implements org.xml.sax.ContentHandler, org.xml.sax.Parser;
public final class XMLReaderFactory;
```

AttributeListImpl

Java 1.4; Deprecated in Java 1.4

org.xml.sax.helpers

This deprecated class is an implementation of the deprecated SAX1 `org.xml.sax.AttributeList` interface. They have been deprecated in favor of the `AttributesImpl` implementation of the SAX2 `org.xml.sax.Attributes` interface.

Object	AttributeListImpl	AttributeList
--------	-------------------	---------------

```
public class AttributeListImpl implements org.xml.sax.AttributeList {
// Public Constructors
    public AttributeListImpl();
    public AttributeListImpl(org.xml.sax.AttributeList atts);
// Public Instance Methods
    public void addAttribute(String name, String type, String value);
    public void clear();
    public void removeAttribute(String name);
    public void setAttributeList(org.xml.sax.AttributeList atts);
// Methods Implementing AttributeList
    public int getLength();                                default:0
    public String getName(int i);
    public String getType(int i);
    public String getType(String name);
    public String getValue(String name);
```


AttributeListImpl

```
    public String getValue(int i);  
}
```

AttributesImpl

Java 1.4

org.xml.sax.helpers

This utility class is a general-purpose implementation of the `Attributes` interface. In addition to implementing all the methods of `Attributes`, it also defines various `set` methods for setting attribute names, values, and types, an `addAttribute()` method for adding a new attribute to the end of the list, a `removeAttribute()` method for removing an attribute from the list, and a `clear()` method for removing all attributes. Also, there is an `AttributesImpl()` constructor that initializes the new `AttributesImpl` object with a copy of a specified `Attributes` object. This class is useful for `XMLFilter` implementations that want to filter the attributes of an element, or for `ContentHandler` implementations that need to make and save a copy of an `Attributes` object for later use.

Object	AttributesImpl	Attributes
--------	----------------	------------

```
public class AttributesImpl implements org.xml.sax.Attributes {  
    // Public Constructors  
    public AttributesImpl();  
    public AttributesImpl(org.xml.sax.Attributes atts);  
    // Public Instance Methods  
    public void addAttribute(String uri, String localName, String qName, String type, String value);  
    public void clear();  
    public void removeAttribute(int index);  
    public void setAttribute(int index, String uri, String localName, String qName, String type, String value);  
    public void setAttributes(org.xml.sax.Attributes atts);  
    public void setLocalName(int index, String localName);  
    public void setQName(int index, String qName);  
    public void setType(int index, String type);  
    public void setURI(int index, String uri);  
    public void setValue(int index, String value);  
    // Methods Implementing Attributes  
    public int getIndex(String qName);  
    public int getIndex(String uri, String localName);  
    public int getLength(); default:0  
    public String getLocalName(int index);  
    public String getQName(int index);  
    public String getType(String qName);  
    public String getType(int index);  
    public String getType(String uri, String localName);  
    public String getURI(int index);  
    public String getValue(int index);  
    public String getValue(String qName);  
    public String getValue(String uri, String localName);  
}
```

DefaultHandler

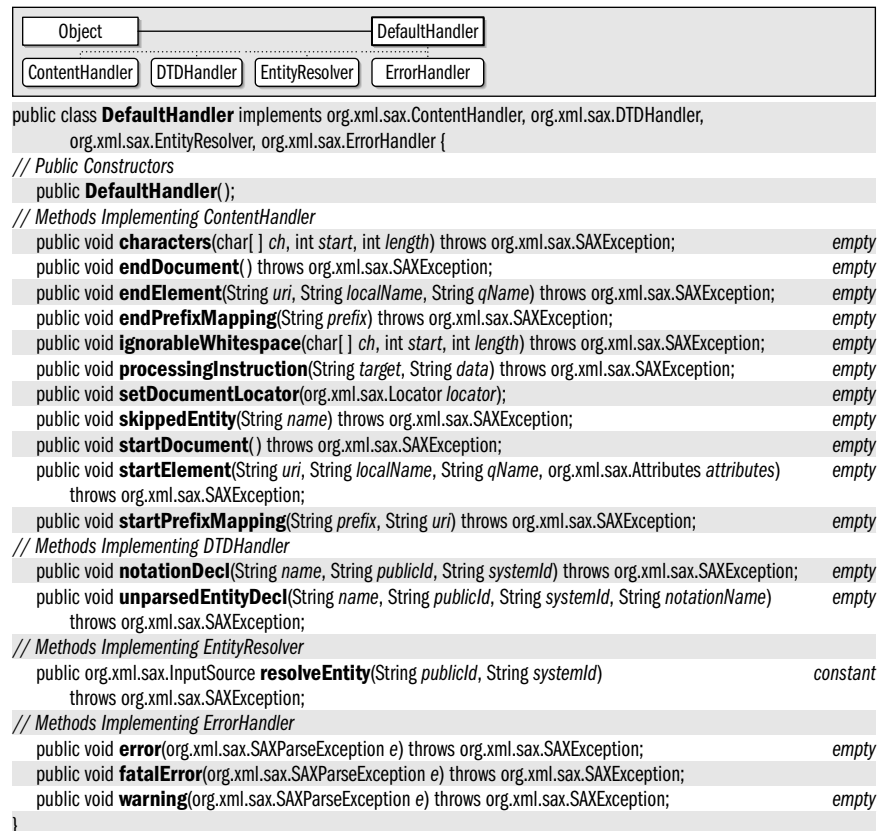
Java 1.4

org.xml.sax.helpers

This helper class implements the four commonly-used SAX handler interfaces from the `org.xml.sax` package and defines stub implementations for all of their methods. It is usually easier to subclass `DefaultHandler` and override the desired methods than it is to implement all of the interfaces (and all of their methods) from scratch. `DefaultHandler`

implements `ContentHandler`, `ErrorHandler`, `EntityResolver` and `DTDHandler`, so you can pass an instance of this class, (or of a subclass you define) to the `setContentHandler()`, `setErrorHandler()`, `setEntityResolver()`, and `setDTDHandler()` methods of an `XMLReader`. You can also pass an instance of a `DefaultHandler` subclass directly to one of the `parse()` methods of a `javax.xml.parsers.SAXParser`. The `SAXParser` will take care of calling the four relevant methods of its internal `XMLReader`.

All but two of the methods of `DefaultHandler` have empty bodies and do nothing. The exceptions are `resolveEntity()` which simply returns null to tell the parser to resolve the entity itself, and `fatalError()` which throws the `SAXParseException` object that is passed to it.



Passed To: `javax.xml.parsers.SAXParser.parse()`

LocatorImpl

Java 1.4

org.xml.sax.helpers

This helper class is a very simple implementation of the `Locator` interface. It defines a copy constructor that create a new `LocatorImpl` object that copies the state of a specified `Locator` object. This constructor is useful because it allows applications to copy the state of a `Locator` and save it for later use.



LocatorImpl

```
public class LocatorImpl implements org.xml.sax.Locator {
// Public Constructors
    public LocatorImpl();
    public LocatorImpl(org.xml.sax.Locator locator);
// Property Accessor Methods (by property name)
    public int getColumnNumber(); Implements:Locator default:0
    public void setColumnNumber(int columnNumber);
    public int getLineNumber(); Implements:Locator default:0
    public void setLineNumber(int lineNumber);
    public String getPublicId(); Implements:Locator default:null
    public void setPublicId(String publicId);
    public String getSystemId(); Implements:Locator default:null
    public void setSystemId(String systemId);
// Methods Implementing Locator
    public int getColumnNumber(); default:0
    public int getLineNumber(); default:0
    public String getPublicId(); default:null
    public String getSystemId(); default:null
}
```

NamespaceSupport

Java 1.4

org.xml.sax.helpers

This utility class exists to help SAX parser implementors handle XML namespaces. It is not commonly used by SAX applications.

```
public class NamespaceSupport {
// Public Constructors
    public NamespaceSupport();
// Public Constants
    public static final String XMLNS; = [quot ]http://www.w3.org/XML/1998/namespace [quot ]
// Public Instance Methods
    public boolean declarePrefix(String prefix, String uri);
    public java.util.Enumeration getDeclaredPrefixes();
    public String getPrefix(String uri);
    public java.util.Enumeration getPrefixes();
    public java.util.Enumeration getPrefixes(String uri);
    public String getURI(String prefix);
    public void popContext();
    public String[] processName(String qName, String[] parts, boolean isAttribute);
    public void pushContext();
    public void reset();
}
```

ParserAdapter

Java 1.4

org.xml.sax.helpers

This adapter class behaves like a SAX2 XMLReader object, but gets its input from the SAX1 Parser object that is passed to the constructor. In order to make this work, it implements the deprecated SAX1 DocumentHandler interface so that it can receive events from the Parser. ParserAdapter provides its own layer of namespace processing to convert a namespace-unaware Parser into a namespace-aware XMLReader. This class is useful when working you are working with a legacy API that supplies a SAX1 Parser object, but want to work with that parser using the SAX2 XMLReader API: to use it, simply pass the Parser object to the ParserAdapter() constructor and use the resulting object as you would use any other XMLReader object.

There is not perfect congruence between the SAX1 and SAX2 APIs, and a `Parser` cannot be perfectly adapted to a `XMLReader`. In particular, a `ParserAdapter` will never call the `skippedEntity()` handler method because the SAX1 `Parser` API does not provide notification of skipped entities. Also, it does not attempt to determine whether two namespace-prefixed attributes of an element actually resolve to the same attribute.

See also `XMLReaderAdapter`, an adapter that works in the reverse direction to make a SAX2 parser behave like a SAX1 parser.



```

public class ParserAdapter implements org.xml.sax.DocumentHandler, org.xml.sax.XMLReader {
// Public Constructors
    public ParserAdapter() throws org.xml.sax.SAXException;
    public ParserAdapter(org.xml.sax.Parser parser);
// Methods Implementing DocumentHandler
    public void characters(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void endDocument() throws org.xml.sax.SAXException;
    public void endElement(String qName) throws org.xml.sax.SAXException;
    public void ignorableWhitespace(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void processingInstruction(String target, String data) throws org.xml.sax.SAXException;
    public void setDocumentLocator(org.xml.sax.Locator locator);
    public void startDocument() throws org.xml.sax.SAXException;
    public void startElement(String qName, org.xml.sax.AttributeList qAtts) throws org.xml.sax.SAXException;
// Methods Implementing XMLReader
    public org.xml.sax.ContentHandler getContentHandler();
    public org.xml.sax.DTDHandler getDTDHandler();
    public org.xml.sax.EntityResolver getEntityResolver();
    public org.xml.sax.ErrorHandler getErrorHandler();
    public boolean getFeature(String name) throws org.xml.sax.SAXNotRecognizedException,
        org.xml.sax.SAXNotSupportedException;
    public Object getProperty(String name) throws org.xml.sax.SAXNotRecognizedException,
        org.xml.sax.SAXNotSupportedException;
    public void parse(String systemId) throws java.io.IOException, org.xml.sax.SAXException;
    public void parse(org.xml.sax.InputSource input) throws java.io.IOException, org.xml.sax.SAXException;
    public void setContentHandler(org.xml.sax.ContentHandler handler);
    public void setDTDHandler(org.xml.sax.DTDHandler handler);
    public void setEntityResolver(org.xml.sax.EntityResolver resolver);
    public void setErrorHandler(org.xml.sax.ErrorHandler handler);
    public void setFeature(String name, boolean state) throws org.xml.sax.SAXNotRecognizedException,
        org.xml.sax.SAXNotSupportedException;
    public void setProperty(String name, Object value) throws org.xml.sax.SAXNotRecognizedException,
        org.xml.sax.SAXNotSupportedException;
}
  
```

ParserFactory

Java 1.4; Deprecated in Java 1.4

org.xml.sax.helpers

This deprecated SAX1 class is a factory for deprecated SAX1 `Parser` objects. New applications should use the SAX2 `XMLReaderFactory` as a factory for SAX2 `XMLReader` objects.

```

public class ParserFactory {
// No Constructor
// Public Class Methods
    public static org.xml.sax.Parser makeParser() throws ClassNotFoundException, IllegalAccessException,
        InstantiationException, NullPointerException;
}
  
```

```

    public static org.xml.sax.Parser makeParser(String className) throws ClassNotFoundException,
        IllegalAccessException, InstantiationException, ClassCastException;
}

```

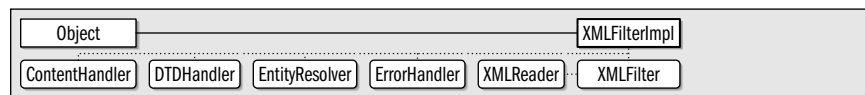
XMLFilterImpl

Java 1.4

org.xml.sax.helpers

This class implements an XMLFilter that does no filtering. You can subclass it to override whatever methods are required to perform the type of filtering you desire.

XMLFilterImpl implements ContentHandler, ErrorHandler, EntityResolver and DTDHandler so that it can receive SAX events from the “parent” XMLReader object. But it also implements the XMLFilter interface, which is an extension of XMLReader, so that it acts as an XMLReader itself, and can send SAX events to the handler objects that are registered on it. Each of the handler methods of this class simply invoke the corresponding method of the corresponding handler that was registered on the filter. The XMLReader methods for getting and setting features and properties simply invoke the corresponding method of the parent XMLReader object. The parse() methods do the same thing: they pass their argument to the corresponding parse() method of the parent reader to start the parsing process.



```

public class XMLFilterImpl implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler,
    org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, org.xml.sax.XMLFilter {
// Public Constructors
    public XMLFilterImpl();
    public XMLFilterImpl(org.xml.sax.XMLReader parent);
// Methods Implementing ContentHandler
    public void characters(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void endDocument() throws org.xml.sax.SAXException;
    public void endElement(String uri, String localName, String qName) throws org.xml.sax.SAXException;
    public void endPrefixMapping(String prefix) throws org.xml.sax.SAXException;
    public void ignorableWhitespace(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void processingInstruction(String target, String data) throws org.xml.sax.SAXException;
    public void setDocumentLocator(org.xml.sax.Locator locator);
    public void skippedEntity(String name) throws org.xml.sax.SAXException;
    public void startDocument() throws org.xml.sax.SAXException;
    public void startElement(String uri, String localName, String qName, org.xml.sax.Attributes atts)
        throws org.xml.sax.SAXException;
    public void startPrefixMapping(String prefix, String uri) throws org.xml.sax.SAXException;
// Methods Implementing DTDHandler
    public void notationDecl(String name, String publicId, String systemId) throws org.xml.sax.SAXException;
    public void unparsedEntityDecl(String name, String publicId, String systemId, String notationName)
        throws org.xml.sax.SAXException;
// Methods Implementing EntityResolver
    public org.xml.sax.InputSource resolveEntity(String publicId, String systemId) throws org.xml.sax.SAXException,
        java.io.IOException;
// Methods Implementing ErrorHandler
    public void error(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException;
    public void fatalError(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException;
    public void warning(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException;
// Methods Implementing XMLFilter
    public org.xml.sax.XMLReader getParent();
    public void setParent(org.xml.sax.XMLReader parent);
}

```

```
// Methods Implementing XMLReader
public org.xml.sax.ContentHandler getContentHandler(); default:null
public org.xml.sax.DTDHandler getDTDHandler(); default:null
public org.xml.sax.EntityResolver getEntityResolver(); default:null
public org.xml.sax.ErrorHandler getErrorHandler(); default:null
public boolean getFeature(String name) throws org.xml.sax.SAXNotRecognizedException,
    org.xml.sax.SAXNotSupportedException;
public Object getProperty(String name) throws org.xml.sax.SAXNotRecognizedException,
    org.xml.sax.SAXNotSupportedException;
public void parse(String systemId) throws org.xml.sax.SAXException, java.io.IOException;
public void parse(org.xml.sax.InputSource input) throws org.xml.sax.SAXException, java.io.IOException;
public void setContentHandler(org.xml.sax.ContentHandler handler);
public void setDTDHandler(org.xml.sax.DTDHandler handler);
public void setEntityResolver(org.xml.sax.EntityResolver resolver);
public void setErrorHandler(org.xml.sax.ErrorHandler handler);
public void setFeature(String name, boolean state) throws org.xml.sax.SAXNotRecognizedException,
    org.xml.sax.SAXNotSupportedException;
public void setProperty(String name, Object value) throws org.xml.sax.SAXNotRecognizedException,
    org.xml.sax.SAXNotSupportedException;
}
```

XMLReaderAdapter

Java 1.4

org.xml.sax.helpers

This adapter class wraps a SAX2 XMLReader object and makes it behave like a SAX1 Parser object. It is useful when working with a legacy API that requires a deprecated Parser object. Create an XMLReaderAdapter by passing an XMLReader to the XMLReaderAdapter() constructor. Then use the resulting object exactly as you would use any other SAX1 Parser object. This class implements ContentHandler so that it can receive SAX events from the XMLReader. But it also implements the Parser interface so that it can have a SAX1 DocumentHandler registered on it. The methods of ContentHandler are implemented to invoke the corresponding methods of the registered DocumentHandler.



```
public class XMLReaderAdapter implements org.xml.sax.ContentHandler, org.xml.sax.Parser {
    // Public Constructors
    public XMLReaderAdapter() throws org.xml.sax.SAXException;
    public XMLReaderAdapter(org.xml.sax.XMLReader xmlReader);
    // Methods Implementing ContentHandler
    public void characters(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void endDocument() throws org.xml.sax.SAXException;
    public void endElement(String uri, String localName, String qName) throws org.xml.sax.SAXException;
    public void endPrefixMapping(String prefix); empty
    public void ignorableWhitespace(char[] ch, int start, int length) throws org.xml.sax.SAXException;
    public void processingInstruction(String target, String data) throws org.xml.sax.SAXException;
    public void setDocumentLocator(org.xml.sax.Locator locator);
    public void skippedEntity(String name) throws org.xml.sax.SAXException; empty
    public void startDocument() throws org.xml.sax.SAXException;
    public void startElement(String uri, String localName, String qName, org.xml.sax.Attributes atts)
        throws org.xml.sax.SAXException;
    public void startPrefixMapping(String prefix, String uri); empty
    // Methods Implementing Parser
}
```

XMLReaderAdapter

```
public void parse(String systemId) throws java.io.IOException, org.xml.sax.SAXException;  
public void parse(org.xml.sax.InputSource input) throws java.io.IOException, org.xml.sax.SAXException;  
public void setDocumentHandler(org.xml.sax.DocumentHandler handler);  
public void setDTDHandler(org.xml.sax.DTDHandler handler);  
public void setEntityResolver(org.xml.sax.EntityResolver resolver);  
public void setErrorHandler(org.xml.sax.ErrorHandler handler);  
public void setLocale(java.util.Locale locale) throws org.xml.sax.SAXException;  
}
```

XMLReaderFactory

Java 1.4

org.xml.sax.helpers

This factory class defines two static factory methods for creating `XMLReader` objects. One method takes the name of a class as its argument. It dynamically loads and instantiates the class, then casts it to an `XMLReader` object. The second factory method takes no arguments; it reads the system property named “org.xml.sax.driver” and uses the value of that property as the name of the class `XMLReader` implementation class to load and instantiate. An application that instantiates its SAX parser using the no-argument method of `XMLReaderFactory` gains a layer of independence from the underlying parser implementation. The end user or system administrator of the system on which the application is deployed can change the parser implementation simply by setting a system property. Note that the `javax.xml.parsers` package provides a similar, but somewhat more useful `SAXParserFactory`.

```
public final class XMLReaderFactory {  
    // No Constructor  
    // Public Class Methods  
    public static org.xml.sax.XMLReader createXMLReader() throws org.xml.sax.SAXException;  
    public static org.xml.sax.XMLReader createXMLReader(String className) throws org.xml.sax.SAXException;  
}
```