Java API for XML Processing

Version 1.1 Public Review 2

Comments to: jsr63-comments@eng.sun.com

Rajiv Mordani

James Duncan Davidson

Scott Boag (Lotus)

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto CA 94303 USA
650 960-1300

November 18, 2000

Java(TM) API for XML Processing (JAXP) Specification ("Specification")
Version: 1.1
Status: Pre-FCS
Release: November 17, 2000

OR OTHER RIGHTS. This document does not represent any commitment to release or implement any portion of the Specification in any product.

THE SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SUN MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

LIMITATION OF LIABILITY
TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF SUN AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend Sun and its licensors from any claims based on your use of the Specification for any purposes other than those of internal evaluation, and from any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

RESTRICTED RIGHTS LEGEND
If this Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

REPORT
You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your evaluation of the Specification ("Feedback"). To the extent that you provide Sun with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Sun a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the Specification and future versions, implementations, and test suites thereof.

SECTION 1 *Overview* ***7***

What is XML? **7**
XML and the Java™ Platform **8**
About this Specification **8**
Who Should Read this Document **8**
Development of this Specification **8**
Report and Contact **9**
Acknowledgements **10**

SECTION 2 *Endorsed Specifications* ***11***

W3C XML 1.0 Recommendation (Second edition) **11**
W3C XML Namespaces 1.0 Recommendation **12**
Simple API for XML Parsing (SAX) 2.0 **12**
Document Object Model (DOM) Level 2 **13**
XSLT 1.0 **13**

SECTION 3 *Plugability Layer* ***15***

SAX Plugability **15**
DOM Plugability **17**
XSLT Plugability **19**
Thread Safety **26**

SECTION 4 *Packages javax.xml.parsers and javax.xml.transform* ***29***

SECTION 5 *Conformance Requirements* ***125***

SECTION 6 *Change History* ***127***

From 1.1 Public Review 1 to 1.1 Public Review 2 **127**
From 1.0 Final Release to 1.1 Public Review **127**
From 1.0 Public Release to 1.0 Final Release **128**

From 1.0 Public Review to 1.0 Public Release **128**

SECTION 7 *Future Directions* ***131***

Updated SAX and DOM Support **131**
Update XSL Plugability Support **131**
Plugability Mechanism Enhancements **132**

SECTION 1

Overview

1.1 What is XML?

XML is the meta language defined by the World Wide Web Consortium (W3C) that can be used to describe a broad range of hierarchical mark up languages. It is a set of rules, guidelines, and conventions for describing structured data in a plain text, editable file. Using a text format instead of a binary format allows the programmer or even an end user to look at or utilize the data without relying on the program that produced it. However the primary producer and consumer of XML data is the computer program and not the end-user.

Like HTML, XML makes use of tags and attributes. Tags are words bracketed by the `'<'` and `'>'` characters and attributes are strings of the form `'name="value"'` that are inside of tags. While HTML specifies what each tag and attribute means, as well as their presentation attributes in a browser, XML uses tags only to delimit pieces of data and leaves the interpretation of the data to the application that uses it. In other words, XML defines only the structure of the document and does not define any of the presentation semantics of that document.

Development of XML started in 1996 leading to a W3C Recommendation in February of 1998. However, the technology is not entirely new. It is based on SGML (Standard Generalized Markup Language) which was developed in the early 1980's and became an ISO standard in 1986. SGML has been widely used for large documentation projects and there is a large community that has experience working with SGML. The designers of XML took the best parts of SGML, used their experience as a guide and produced a technology that is just as powerful as SGML, but much simpler and easier to use.

XML-based documents can be used in a wide variety of applications including vertical markets, e-commerce, business-to-business communication, and enterprise application messaging.

1.2 XML and the Java™ Platform

In many ways, XML and the Java Platform are a partnership made in heaven. XML defines a cross platform data format and Java provides a standard cross platform programming platform. Together, XML and Java technologies allow programmers to apply Write Once, Run Anywhere™ fundamentals to the processing of data and documents generated by both Java based programs and non-Java based programs.

1.3 About this Specification

This document describes the Java API for XML Processing, Version 1.1. This version of the specification introduces basic support for parsing and manipulating XML documents through a standardized set of Java Platform APIs.

When this specification is final there will be a Reference Implementation which will demonstrate the capabilities of this API and will provide an operational definition of the specification. A Technology Compatibility Kit (TCK) will also be available that will verify whether an implementation of this specification is compliant. These are required as per the Java Community Process 2.0 (JCP 2.0).

1.4 Who Should Read this Document

This specification is intended for use by:

- Parser Developers wishing to implement this version of the specification in their parser.
- Application Developers who use the APIs described in this specification and wish to have a more complete understanding of the API.

This specification is not a tutorial or a user's guide to XML, DOM, SAX or XSLT. Familiarity with these technologies and specifications on the part of the reader is assumed.

1.5 Development of this Specification

This specification was developed in accordance with the Java Community Process 2.0. It was developed under the authorization of Java Specification Request 63. More information about the Java Community Process can be found at:

`http://java.sun.com/jcp/`

The specific information contained in Java Specification Request 63 can be found at:

`http://java.sun.com/aboutJava/communityprocess/jsr/jsr_063_jaxp.html`

The expert group who contributed to this specification is composed of individuals from a number of companies. These individuals are:

- James Duncan Davidson (co-lead), Sun Microsystems
- Rajiv Mordani (co-lead), Sun Microsystems
- Scott Boag, Lotus.
- Kelvin Lawrence, IBM
- Jeff Mischinkinsky, Persistence
- Todd Karakashain, BEA
- Tom Reilly, Allaire
- Tom Bates, Informix
- Miles Sabin, CromwellMedia
- Wolfram Kaiser, POET
- Paul Boutros, eBusiness Technologies
- Pier Fumagalli, Apache Software Foundation
- Stefano Mazzocchi, Apache Software Foundation
- Takuki Kamiya, Fujitsu Ltd

1.6 Report and Contact

Your comments on this specification are welcome and appreciated. Without your comments, the specifications developed under the auspices of the Java Community Process would not serve your needs as well. To comment on this specification, please send email to:

`jsr63-comments@eng.sun.com`

You can stay current with Sun's Java Platform related activities, as well as information on our `xml-interest` and `xml-announce` mailing lists, at our website located at:

`http://java.sun.com/xml/`

1.7 Acknowledgements

Many individuals and companies have given their time and talents to make this specification, or the specifications that this specification relies upon, a reality. The author of this specification would like to thank (in no particular order):

- David Megginson and the XML-DEV community who developed the SAX API
- Mikael Staldal, Michael Kay, and the contributors to the original TrAX mailing list.
- The W3C DOM Working Group chaired by Lauren Wood
- The JSR-63 Expert Group listed above
- Graham Hamilton, Mark Hapner, Eduardo Pelegri-Lopart, Connie Weiss, Jim Driscoll, Edwin Goei, Costin Manolache, Mark Reinhold, Bill Shannon, Vivek Nagar, Janet Breuer, Jeff Jackson and Will Iverson all of whom work at Sun Microsystems and whose talents have all reflected upon the development of this API.

SECTION 2

Endorsed Specifications

This specification endorses and builds upon several external specifications. Each specification endorsed by this document is called out together with the exact version of the specification and its publicly accessible location. All of these standards have conformance tests provided in the Technology Compatibility Kit available for this specification.

2.1 W3C XML 1.0 Recommendation (Second edition)

The W3C XML 1.0 Recommendation specifies the core XML syntax by subsetting the existing, widely used international SGML[1] text processing standard. It is a product of the W3C XML Activity, details of which can be found at:

`http://www.w3.org/XML/`

The XML 1.0 Recommendation (second edition) can be located at:

http://www.w3.org/TR/2000/REC-xml-20001006

This specification includes by reference the XML 1.0 Recommendation (second edition) in its entirety for the purposes of defining the XML language manipulated by the APIs defined herein.

1. Standard Generalized Markup Language, ISO 8879:1986(E) as amended and corrected.

2.2 W3C XML Namespaces 1.0 Recommendation

The W3C XML Namespaces Recommendation defines the syntax and semantics for XML structures required to be distinct from other XML markup. In particular, it defines a mechanism whereby a set of XML markup may have a distinguishing "namespace" associated with it, and the responsibility of XML parser in handling and exposing such namespace information.

The XML Namespaces 1.0 Recommendation is located at:

`http://www.w3.org/TR/1999/REC-xml-names-19990114/`

This specification includes by reference the XML Namespaces 1.0 Recommendation in its entirety.

2.3 Simple API for XML Parsing (SAX) 2.0

The Simple API for XML (SAX) is a public domain API developed cooperatively by the members of the XML-DEV mailing list. It provides an event-driven interface to the process of parsing an XML document.

An event driven interface provides a mechanism for a "callback" notifications to application’s code as the underlying parser recognizes XML syntactic constructions in the document.

The SAX 2.0 API is located at:

`http://www.megginson.com/SAX/index.html`

The SAX 2 extensions is located at:

`http://www.megginson.com/Software/sax2-ext-1.0.zip`

`The details of the XML-DEV mailing list can be found at`

`http://xml.org/xml-dev/index.shtml`

This specification includes by reference the SAX 2.0 API and the SAX2 extensions in its entirety.

The API packages included by reference are:

- `org.xml.sax`
- `org.xml.sax.helpers`
- `org.xml.sax.ext`

2.4 Document Object Model (DOM) Level 2

The Document Object Model (DOM) is a set of interfaces defined by the W3C DOM Working Group. It describes facilities for a programmatic representation of a parsed XML (or HTML) document. The DOM Level 2 specification defines these interfaces using Interface Definition Language (IDL) in a language independent fashion and also includes a Java Language binding.

The DOM Level 2 Core Recommendation is located at:

http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/

This specification includes by reference both the abstract semantics described for the DOM Level 2 Core Recommendation interfaces and the associated Java Language binding. It does not include the optional extensions defined by the DOM working group. The API package included by this specification is:

- `org.w3c.dom`

2.5 XSLT 1.0

The XSL Transformations (XSLT) describes a language for transforming XML documents into other XML documents or other text output. It was defined by the W3C XSL Working group.

The XSLT 1.0 Recommendation is located at:

```
http://www.w3.org/TR/1999/REC-xslt-19991116
```

This specification includes by reference the XSLT 1.0 specification in its entirety.

SECTION 3

Plugability Layer

The endorsed APIs provide broad and useful functionality. However, the use of a SAX or a DOM parser typically requires knowledge of the specific implementation of the parser. Providing the functionality of the endorsed APIs in the Java Platform, while allowing choice of the implementation of the parser, requires a Plugability layer.

This section of the specification defines a Plugability mechanism to allow a compliant SAX or DOM parser to be used through the abstract `javax.xml.parsers` and `javax.xml.transoform` API.

3.1 SAX Plugability

The SAX Plugability classes allow an application programmer to provide an implementation of the `org.xml.sax.DefaultHandler` API to a `SAXParser` implementation and parse XML documents. As the parser processes the XML document, it will call methods on the provided `DefaultHandler`.

In order to obtain a `SAXParser` instance, an application programmer first obtains an instance of a `SAXParserFactory`. The `SAXParserFactory` instance is obtained via the static `newInstance` method of the `SAXParserFactory` class.

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 JAVA_HOME (the parent directory where jdk is installed)/lib/jaxp.properties for a property file that contains the name of the implementation class keyed on the same value as the system property defined above.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file `META-INF/services/javax.xml.parsers.SAXParserFactory` in jars available to the runtime.
- Platform default `SAXParserFactory` instance.

If the `SAXParserFactory` implementation class cannot be loaded or instantiated at runtime, a `FactoryConfigurationError` is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

The instance of `SAXParserFactory` can optionally be configured by the application programmer to provide parsers that are namespace aware, or validating, or both. These settings are made using the `setNamespaceAware` and `setValidating` methods of the factory. The application programmer can then obtain a `SAXParser` implementation instance from the factory. If the factory cannot provide a parser configured as set by the application programmer, then a `ParserConfigurationException` is thrown.

3.1.1 Examples

The following is a simple example of how to parse XML content from a URL:

```
SAXParser parser;
DefaultHandler handler = new MyApplicationParseHandler();
SAXParserFactory factory = SAXParserFactory.newInstance();
try {
    parser = factory.newSAXParser();
    parser.parse("http://myserver/mycontent.xml", handler);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

The following is an example of how to configure a SAX parser to be namespace aware and validating:

```
SAXParser parser;
DefaultHandler handler = new MyApplicationParseHandler();
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setNamespaceAware(true);
factory.setValidating(true);
try {
```

```
    parser = factory.newSAXParser();
    parser.parse("http://myserver/mycontent.xml", handler);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

An example of how one could pass the System property as a command line option is shown below

```
java -Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
user.parserApp.
```

3.2 DOM Plugability

The DOM plugability classes allow a programmer to parse an XML document and obtain an `org.w3c.dom.Document` object from a `DocumentBuilder` implementation which wraps an underlying DOM implementation.

In order to obtain a `DocumentBuilder` instance, an application programmer first obtains an instance of a `DocumentBuilderFactory`. The `DocumentBuilderFactory` instance is obtained via the static `newInstance` method of the `DocumentBuilderFactory` class.

This method uses the following ordered lookup procedure to determine the `DocumentBuilderFactory` implementation class to load:

- Use the `javax.xml.parsers.DocumentBuilderFactory` system property
- Use the JAVA_HOME (the parent directory where jdk is installed)/lib/jaxp.properties for a property file that contains the name of the implementation class keyed on the same value as the system property defined above.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file `META-INF/services/javax.xml.parsers.DocumentBuilderFactory` in jars available to the runtime.
- Platform default `DocumentBuilderFactory` instance.

If the `DocumentBuilderFactory` implementation class cannot be loaded or instantiated at runtime, a `FactoryConfigurationError` is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

The instance of `DocumentBuilderFactory` can optionally be configured by the application programmer to provide parsers that are namespace aware or validating, or both. These settings are made using the `setNamespaceAware` and `setValidating` methods of the factory. The application programmer can then obtain a `DocumentBuilder` implementation instance from the factory. If the factory cannot provide a parser configured as set by the application programmer, then a `ParserConfigurationException` is thrown.

3.2.1 Reliance on SAX API

The DocumentBuilder reuses several classes from the SAX API. This does not mean that the implementor of the underlying DOM implementation must use a SAX parser to parse the XML content, only that the implementation communicate with the application using these existing and defined APIs.

3.2.2 Examples

The following is a simple example of how to parse XML content from a URL:

```
DocumentBuilder builder;
DocumentBuilderFactory factory =
    DocumentBuilderFactory.newInstance();
String location = "http://myserver/mycontent.xml";
try {
    builder = factory.newDocumentBuilder();
    Document document = builder.parse(location);
} catch (SAXException se) {
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

The following is an example of how to configure a factory to produce parsers to be namespace aware and validating:

```
DocumentBuilder builder;
DocumentBuilderFactory factory =
    DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
factory.setValidating(true);
String location = "http://myserver/mycontent.xml";
try {
    builder = factory.newDocumentBuilder();
    Document document = builder.parse(location);
} catch (SAXException se) {
```

```
    // handle error
} catch (IOException ioe) {
    // handle error
} catch (ParserConfigurationException pce) {
    // handle error
}
```

An example of how one could pass the System property as a command line option is shown below

```
java -Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilder-
FactoryImpl user.parserApp.
```

3.3 XSLT Plugability

The XSLT Plugability classes allow an application programmer to obtain a `Transform` object that is based on a specific XSLT stylesheet from a `TransformerFactory` implementation. In order to obtain a `Transform`er object, a programmer first obtains an instance of the `TransformerFactory`. The `TransformerFactory` instance is obtained via the static newInstance method of the `TransformFactory` class.

This method uses the following ordered lookup procedure to determine the `TransformerFactory` implementation class to load:

- Use the `javax.xml.transform.TransformerFactory` system property
- Use the JAVA_HOME (the parent directory where jdk is installed)/lib/jaxp.properties for a property file that contains the name of the implementation class keyed on the same value as the system property defined above.
- Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for the classname in the file `META-INF/services/javax.xml.transform.TransformerFactory` in jars available to the runtime.
- Platform default `TransformFactory` instance.

If the `TransformerFactory` implementation class cannot be loaded or instantiated at runtime, a `TFactoryConfigurationError` is thrown. This error message should contain a descriptive explanation of the problem and how the user can resolve it.

3.3.1 Examples

The following is a simple example of how to transform XML content:

```
Transformer transformer;
```

```
TransformerFactory factory = TransformerFactory.newInstance();
String stylesheet = "file:///home/user/mystylesheet.xsl";
String sourceId = "file:///home/user/sourcefile.xml";
try {
    transformer = factory.newTransform(
                          new StreamSource(stylesheet));
    transform.transform(new StreamSource(sourceId),
                        new StreamSource(System.out));
} catch (Exception e) {
    // handle error
}
```

The following example illustrates the serialization of a DOM node to an XML stream.

```
TransformerFactory tfactory = TransformerFactory.newInstance();
Transformer serializer = tfactory.newTransformer();
Properties oprops = new Properties();
oprops.put(“method”, “html”);
oprops.put(“indent-amount”, “2”);
serializer.setOutputProperties(oprops);
serializer.transform(new DOMSource(doc),
                     new StreamResult(System.out));
Exceptions and Error Reporting
```

The following example illustrates the use of the URI resolver to resolve URIs to DOM nodes, in a transformation whose input is totally DOM based.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

if (tfactory.getFeature(DOMSource.FEATURE) && tfactory.getFeature(StreamResult.FEATURE))
{
  DocumentBuilderFactory dfactory =
                                    DocumentBuilderFactory.newInstance();
  dfactory.setNamespaceAware(true); // Always, required for XSLT
  DocumentBuilder docBuilder = dfactory.newDocumentBuilder();

  // Set up to resolve URLs that correspond to our inc1.xsl,
  // to a DOM node. Use an anonymous class for the URI resolver.
  final Node xslInc1 = docBuilder.parse("xsl/inc1/inc1.xsl");
  final Node xslInc2 = docBuilder.parse("xsl/inc2/inc2.xsl");
  tfactory.setURIResolver(new URIResolver() {
    public Source resolve(String href, String base)
    throws TransformerException
    {
    // ignore base.
    return (href.equals("inc1/inc1.xsl"))
    ? new DOMSource(xslInc1) :
      (href.equals("inc2/inc2.xsl"))
      ? new DOMSource(xslInc2) : null;
    }});

  // The TransformerFactory will call the anonymous URI
  // resolver set above when it encounters
  // <xsl:include href="inc1/inc1.xsl"/>
  Templates templates
    = tfactory.newTemplates(new DOMSource(docBuilder.parse(xslID), xslID));

  // Get a transformer from the templates.
  Transformer transformer = templates.newTransformer();

  // Set up to resolve URLs that correspond to our foo2.xml, to
  // a DOM node.  Use an anonymous class for the URI resolver.
  // Be sure to return the same DOM tree every time for the
  // given URI.
  final Node xmlSubdir1Foo2Node = docBuilder.parse("xml/subdir1/foo2.xml");
  transformer.setURIResolver(new URIResolver() {
    public Source resolve(String href, String base)
    throws TransformerException
    {
    // ignore base because we're lazy, or we don't care.
    return (href.equals("subdir1/foo2.xml"))
    ? new DOMSource(xmlSubdir1Foo2Node) : null;
    }});

  // Now the transformer will call our anonymous URI resolver
  // when it encounters the document('subdir1/foo2.xml') invocation.
  transformer.transform(new DOMSource(docBuilder.parse(sourceID), sourceID),
                        new StreamResult(System.out));
}
```

The following example performs a transformation using DOM nodes as input for the TransformerFactory, as input for the Transformer, and as the output of the transformation.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

// Make sure the TransformerFactory supports the DOM feature.
if (tfactory.getFeature(DOMSource.FEATURE) && tfactory.getFeature(DOMResult.FEATURE))
{
  // Use javax.xml.parsers to create our DOMs.
  DocumentBuilderFactory dfactory = DocumentBuilderFactory.newInstance();
  dfactory.setNamespaceAware(true); // do this always for XSLT
  DocumentBuilder docBuilder = dfactory.newDocumentBuilder();

  // Create the Templates from a DOM.
  Node xslDOM = docBuilder.parse(xslID);
  DOMSource dsource = new DOMSource(xslDOM, xslID);
  Templates templates = tfactory.newTemplates(dsource);

  // Create the source tree in the form of a DOM.
  Node sourceNode = docBuilder.parse(sourceID);

  // Create a DOMResult that the transformation will fill in.
  DOMResult dresult = new DOMResult();

  // And transform from the source DOM tree to a result DOM tree.
  Transformer transformer = templates.newTransformer();
  transformer.transform(new DOMSource(sourceNode, sourceID), dresult);

  // The root of the result tree may now be obtained from
  // the DOMResult object.
  Node out = dresult.getNode();

  // Serialize it to System.out for diagnostics.
  Transformer serializer = tfactory.newTransformer();
  serializer.transform(new DOMSource(out), new StreamResult(System.out));
}
```

The following code fragment illustrates the use of the SAXSource and SAXResult objects.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

// Does this factory support SAX features?
if (tfactory.getFeature(SAXSource.FEATURE) && tfactory.getFeature(SAXResult.FEATURE))
{
  // Get a transformer.
  Transformer transformer
    = tfactory.newTransformer(new StreamSource(xslID));

  // Create an reader for reading.
  XMLReader reader = XMLReaderFactory.createXMLReader();

  transformer.transform(new SAXSource(reader, new InputSource(sourceID)),
                        new SAXResult(new ExampleContentHandler()));
}
```

The following illustrates the feeding of SAX events from an `org.xml.sax.XMLReader` to a Transformer.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE))
{
  // If so, we can safely cast.
  SAXTransformerFactory stfactory = ((SAXTransformerFactory) tfactory);

  // A TransformerHandler is a ContentHandler that will listen for
  // SAX events, and transform them to the result.
  TransformerHandler handler
    = stfactory.newTransformerHandler(new StreamSource(xslID));

  // Set the result handling to be a serialization to System.out.
  handler.setResult(new StreamResult(System.out));

  handler.getTransformer().setParameter(“a-param”,
                                        “hello to you!”);

  // Create a reader, and set it's content handler to be the TransformerHandler.
  XMLReader reader = XMLReaderFactory.createXMLReader();
  reader.setContentHandler(handler);

  // It's a good idea for the parser to send lexical events.
  // The TransformerHandler is also a LexicalHandler.
  reader.setProperty(“http://xml.org/sax/properties/lexical-handler”, handler);

  // Parse the source XML, and send the parse events to the TransformerHandler.
  reader.parse(sourceID);
}
```

The following code fragment illustrates the creation of a Templates object from SAX2 events sent from an XMLReader.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE))
{
  // If so, we can safely cast.
  SAXTransformerFactory stfactory = ((SAXTransformerFactory) tfactory);

  // Have the factory create a special ContentHandler that will
  // create a Templates object.
  TemplatesHandler handler = stfactory.newTemplatesHandler();

  // If you don't do this, the TemplatesHandler won't know how to
  // resolve relative URLs.
  handler.setSystemId(xslID);

  // Create a reader, and set it's content handler to be the TemplatesHandler.
  XMLReader reader = XMLReaderFactory.createXMLReader();
  reader.setContentHandler(handler);

  // Parse the source XML, and send the parse events to the TemplatesHandler.
  reader.parse(xslID);

  // Get the Templates reference from the handler.
  Templates templates = handler.getTemplates();

  // Ready to transform.
  Transformer transformer = templates.newTransformer();
  transformer.transform(new StreamSource(sourceID), new StreamResult(System.out));
}
```

The following illustrates several transformations chained together. Each filter points to a parent `org.xml.sax.XMLReader`,and the final transformation is caused by invoking `org.xml.sax.XMLReader#parse` on the final reader in the chain.

```
TransformerFactory tfactory = TransformerFactory.newInstance();

// Does this factory support SAX features?
if (tfactory.getFeature(SAXTransformerFactory.FEATURE))
{
  Templates stylesheet1 = tfactory.newTemplates(new StreamSource(xslID_1));
  Transformer transformer1 = stylesheet1.newTransformer();

  SAXTransformerFactory stf = (SAXTransformerFactory)tfactory;
  XMLReader reader = XMLReaderFactory.createXMLReader();

  XMLFilter filter1 = stf.newXMLFilter(new StreamSource(xslID_1));
  XMLFilter filter2 = stf.newXMLFilter(new StreamSource(xslID_2));
  XMLFilter filter3 = stf.newXMLFilter(new StreamSource(xslID_3));

  // transformer1 will use a SAX parser as it's reader.
  filter1.setParent(reader);

  // transformer2 will use transformer1 as it's reader.
  filter2.setParent(filter1);

  // transform3 will use transform2 as it's reader.
  filter3.setParent(filter2);

  filter3.setContentHandler(new ExampleContentHandler());
  // filter3.setContentHandler(new org.xml.sax.helpers.DefaultHandler());

  // Now, when you call transformer3 to parse, it will set
  // itself as the ContentHandler for transform2, and
  // call transform2.parse, which will set itself as the
  // content handler for transform1, and call transform1.parse,
  // which will set itself as the content listener for the
  // SAX parser, and call parser.parse(new InputSource(“xml/foo.xml”)).
  filter3.parse(new InputSource(sourceID));
}
```

The following code fragment illustrates the use of the stream Source and Result objects.

```
// Create a TransformerFactory instance.
TransformerFactory tfactory = TransformerFactory.newInstance();

InputStream xslIS = new BufferedInputStream(new FileInputStream(xslID));
StreamSource xslSource = new StreamSource(xslIS);
// Note that if we don't do this, relative URLs cannot be resolved correctly!
xslSource.setSystemId(xslID);

// Create a transformer for the stylesheet.
Transformer transformer = tfactory.newTransformer(xslSource);

InputStream xmlIS = new BufferedInputStream(new FileInputStream(sourceID));
StreamSource xmlSource = new StreamSource(xmlIS);
// Note that if we don't do this, relative URLs cannot be resolved correctly!
xmlSource.setSystemId(sourceID);

// Transform the source XML to System.out.
transformer.transform( xmlSource, new StreamResult(System.out));
```

An example of how one could pass the System property as a command line option is shown below

```
java -Djavax.xml.transform.TransformerFactory=org.apache.xerces.jaxp.TransformerFactory-
Impl user.parserApp.
```

3.4 Thread Safety

Implementations of the `SAXParser`, `DocumentBuilder` and `Transformer` abstract classes are not expected to be thread safe by this specification. This means that application programmers should not expect to be able to use the same instance of a `SAXParser`, `DocumentBuilder or Transformer` in more than one thread at a time without side effects. If a programmer is creating a multi-threaded application, they should make sure that only one thread has access to any given `SAXParser`, `DocumentBuilder or Transformer` instance.

Configuration of a `SAXParserFactory`, `DocumentBuilderFactory or TransformerFactory` is also not expected to be thread safe. This means that an application programmer should not allow a `SAXParserFactory` or `DocumentBuilderFactory` to have its `setNamespaceAware` or `setValidating` methods from more than one thread.

It is expected that the `newSAXParser` method of a `SAXParserFactory` implementation, the `newDocumentBuilder` method of a `DocumentBuilderFactory and the newTransformer method of a TransformerFactory` will be thread safe without side effects. This means that an application programmer should expect to be able to create parser instances in multiple threads at once from a shared factory without side effects or problems.

SECTION 4

Packages javax.xml.parsers and javax.xml.transform

This section defines the API of the `javax.xml.parsers`, `javax.xml.transform`, `javax.xml.transform.dom` `javax.xml.transform.sax` and `javax.xml.transform.stream` packages.

Package
javax.xml.parsers

Description

Provides classes allowing the processing of XML documents. Two types of pluggable parsers are supported:

- SAX (Simple API for XML)
- DOM (Document Object Model)

Class Summary

Classes

`DocumentBuilder`	Defines the API to obtain DOM Document instances from an XML document.
`DocumentBuilderFactory`	Defines a factory API that enables applications to obtain a parser that produces DOM object trees from XML documents.
`SAXParser`	Defines the API that wraps an `org.xml.sax.XMLReader` implementation class.
`SAXParserFactory`	Defines a factory API that enables applications to configure and obtain a SAX based parser to parse XML documents.

Exceptions

`ParserConfigurationException`	Thrown when a problem with configuration with the Parser Factories exists.

Errors

`FactoryConfigurationError`	Indicates a serious configuration error.

javax.xml.parsers
DocumentBuilder

Syntax

```
public abstract class DocumentBuilder
```

```
java.lang.Object
  |
  +--javax.xml.parsers.DocumentBuilder
```

Description

Defines the API to obtain DOM Document instances from an XML document. Using this class, an application programmer can obtain a `org.w3c.dom.Document` from XML.

An instance of this class can be obtained from the `DocumentBuilderFactory.newDocumentBuilder` 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.

Note that this class reuses several classes from the SAX API. This does not require that the implementor of the underlying DOM implmenetation use a SAX parser to parse XML document into a `Document`. It merely requires that the implementation communicate with the application using these existing APIs.

Since: JAXP 1.0

Member Summary

Constructors

`DocumentBuilder()`	

Methods

`isNamespaceAware()`	Indicates whether or not this parser is configured to understand namespaces.
`isValidating()`	Indicates whether or not this parser is configured to validate XML documents.
`newDocument()`	Obtain a new instance of a DOM Document object to build a DOM tree with.
`parse(File)`	Parse the content of the given file as an XML document and return a new DOM Document object.
`parse(InputSource)`	Parse the content of the given input source as an XML document and return a new DOM Document object.
`parse(InputStream)`	Parse the content of the given InputStream as an XML document and return a new DOM Document object.
`parse(InputStream, String)`	Parse the content of the given InputStream as an XML document and return a new DOM Document object.
`parse(String)`	Parse the content of the given URI as an XML document and return a new DOM Document object.
`setEntityResolver(EntityResolver)`	Specify the `EntityResolver` to be used to resolve entities present in the XML document to be parsed.
`setErrorHandler(ErrorHandler)`	Specify the `ErrorHandler` to be used to resolve entities present in the XML document to be parsed.

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

Constructors

DocumentBuilder()

```
protected DocumentBuilder()
```

Methods

isNamespaceAware()

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

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

isValidating()

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

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

newDocument()

```
public abstract org.w3c.dom.Document newDocument()
```

Obtain a new instance of a DOM Document object to build a DOM tree with.

parse(File)

```
public org.w3c.dom.Document parse(java.io.File f)
```

Parse the content of the given file as an XML document and return a new DOM Document object.

Parameters:
`f` - The file containing the XML to parse

Throws: `IOException` - If any IO errors occur.

`SAXException` - If any parse errors occur.

`IllegalArgumentException` - If the file is null

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputSource)

```
public abstract org.w3c.dom.Document parse(org.xml.sax.InputSource is)
```

Parse the content of the given input source as an XML document and return a new DOM Document object.

Parameters:
`is` - InputSource containing the content to be parsed.

Throws: `IOException` - If any IO errors occur.

`SAXException` - If any parse errors occur.

`IllegalArgumentException` - If the InputSource is null

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputStream)

```
public org.w3c.dom.Document parse(java.io.InputStream is)
```

Parse the content of the given InputStream as an XML document and return a new DOM Document object.

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

Throws: `IOException` - If any IO errors occur.

`SAXException` - If any parse errors occur.

`IllegalArgumentException` - If the InputStream is null

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputStream, String)

```
public org.w3c.dom.Document parse(java.io.InputStream is, java.lang.String systemId)
```

Parse the content of the given InputStream as an XML document and return a new DOM Document object.

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

`systemId` - Provide a base for resolving relative URIs.

Throws: `IOException` - If any IO errors occur.

`SAXException` - If any parse errors occur.

`IllegalArgumentException` - If the InputStream is null

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

parse(String)

```
public org.w3c.dom.Document parse(java.lang.String uri)
```

Parse the content of the given URI as an XML document and return a new DOM Document object.

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

Throws: `IOException` - If any IO errors occur.

`SAXException` - If any parse errors occur.

`IllegalArgumentException` - If the URI is null

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

setEntityResolver(EntityResolver)

```
public abstract void setEntityResolver(org.xml.sax.EntityResolver er)
```

Specify the `EntityResolver` to be used to resolve entities present in the XML document to be parsed. Setting this to `null` will result in the underlying implementation using it's own default implementation and behavior.

setErrorHandler(ErrorHandler)

```
public abstract void setErrorHandler(org.xml.sax.ErrorHandler eh)
```

Specify the `ErrorHandler` to be used to resolve entities present in the XML document to be parsed. Setting this to `null` will result in the underlying implementation using it's own default implementation and behavior.

javax.xml.parsers
DocumentBuilderFactory

Syntax

```
public abstract class DocumentBuilderFactory

java.lang.Object
  |
  +--javax.xml.parsers.DocumentBuilderFactory
```

Description

Defines a factory API that enables applications to obtain a parser that produces DOM object trees from XML documents.

Since: JAXP 1.0

Member Summary

Constructors

DocumentBuilderFactory()	

Methods

getAttribute(String)	Allows the user to retrieve specific attributes on the underlying implementation.
isCoalescing()	Indicates whether or not the factory is configured to produce parsers which converts CDATA nodes to Text nodes and appends it to the adjacent (if any) Text node.
isExpandEntityReferences()	Indicates whether or not the factory is configured to produce parsers which expand entity reference nodes.
isIgnoringComments()	Indicates whether or not the factory is configured to produce parsers which ignores comments.
isIgnoringElementContentWhitespace()	Indicates whether or not the factory is configured to produce parsers which ignore ignorable whitespace in element content.
isNamespaceAware()	Indicates whether or not the factory is configured to produce parsers which are namespace aware.
isValidating()	Indicates whether or not the factory is configured to produce parsers which validate the XML content during parse.
newDocumentBuilder()	Creates a new instance of a DocumentBuilder using the currently configured parameters.
newInstance()	Obtain a new instance of a `DocumentBuilderFactory`.
setAttribute(String, Object)	Allows the user to set specific attributes on the underlying implementation.
setCoalescing(boolean)	Specifies that the parser produced by this code will convert CDATA nodes to Text nodes and append it to the adjacent (if any) text node.
setExpandEntityReferences(boolean)	Specifies that the parser produced by this code will expand entity reference nodes.
setIgnoringComments(boolean)	Specifies that the parser produced by this code will ignore comments.
setIgnoringElementContentWhitespace(boolean)	Specifies that the parsers created by this factory must eleminate whitespace in element content (sometimes known loosely as 'ignorable whitespace') when parsing XML documents (see XML Rec 2.10).

Member Summary	
setNamespaceAware(boolean)	Specifies that the parser produced by this code will provide support for XML namespaces.
setValidating(boolean)	Specifies that the parser produced by this code will validate documents as they are parsed.

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

Constructors

DocumentBuilderFactory()

```
protected DocumentBuilderFactory()
```

Methods

getAttribute(String)

```
public abstract java.lang.Object getAttribute(java.lang.String name)
```

Allows the user to retrieve specific attributes on the underlying implementation.

Parameters:
`name` - The name of the attribute.

Returns: value The value of the attribute.

Throws: `IllegalArgumentException` - thrown if the underlying implementation doesn't recognize the attribute.

isCoalescing()

```
public boolean isCoalescing()
```

Indicates whether or not the factory is configured to produce parsers which converts CDATA nodes to Text nodes and appends it to the adjacent (if any) Text node.

isExpandEntityReferences()

```
public boolean isExpandEntityReferences()
```

Indicates whether or not the factory is configured to produce parsers which expand entity reference nodes.

---

isIgnoringComments()

```
public boolean isIgnoringComments()
```

Indicates whether or not the factory is configured to produce parsers which ignores comments.

---

isIgnoringElementContentWhitespace()

```
public boolean isIgnoringElementContentWhitespace()
```

Indicates whether or not the factory is configured to produce parsers which ignore ignorable whitespace in element content.

---

isNamespaceAware()

```
public boolean isNamespaceAware()
```

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

---

isValidating()

```
public boolean isValidating()
```

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

---

newDocumentBuilder()

```
public abstract DocumentBuilder newDocumentBuilder()
```

Creates a new instance of a DocumentBuilder using the currently configured parameters.

Throws: `ParserConfigurationException` - if a DocumentBuilder cannot be created which satisfies the configuration requested

---

newInstance()

```
public static DocumentBuilderFactory newInstance()
```

Obtain a new instance of a `DocumentBuilderFactory`. 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.DocumentBuilderFactory”. 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.

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

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

setAttribute(String, Object)

```
public abstract void setAttribute(java.lang.String name, java.lang.Object value)
```

Allows the user to set specific attributes on the underlying implementation.

Parameters:

`name` - The name of the attribute.

`value` - The value of the attribute.

Throws: `IllegalArgumentException` - thrown if the underlying implementation doesn't recognize the attribute.

setCoalescing(boolean)

```
public void setCoalescing(boolean coalescing)
```

Specifies that the parser produced by this code will convert CDATA nodes to Text nodes and append it to the adjacent (if any) text node.

setExpandEntityReferences(boolean)

```
public void setExpandEntityReferences(boolean expandEntityRef)
```

Specifies that the parser produced by this code will expand entity reference nodes.

setIgnoringComments(boolean)

```
public void setIgnoringComments(boolean ignoreComments)
```

Specifies that the parser produced by this code will ignore comments.

setIgnoringElementContentWhitespace(boolean)

```
public void setIgnoringElementContentWhitespace(boolean whitespace)
```

Specifies that the parsers created by this factory must eleminate whitespace in element content (sometimes known loosely as 'ignorable whitespace') when parsing XML documents (see XML Rec 2.10). Note that only whitespace which is directly contained within an element content that has an element only content model (see XML Rec 3.2.1) will be eliminated. Due to reliance on the content model this setting requires the parser to be in validating mode.

setNamespaceAware(boolean)

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

Specifies that the parser produced by this code will provide support for XML namespaces.

setValidating(boolean)

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

Specifies that the parser produced by this code will validate documents as they are parsed.

javax.xml.parsers
FactoryConfigurationError

Syntax

```
public class FactoryConfigurationError extends java.lang.Error
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Error
              |
              +--javax.xml.parsers.FactoryConfigurationError
```

All Implemented Interfaces: java.io.Serializable

Description

Thrown when a problem with configuration with the Parser Factories exists. This error will typically be thrown when the class of a parser factory specified in the system properties cannot be found or instantiated.

Since: JAXP 1.0

Member Summary

Constructors

FactoryConfiguration-Error()	Create a new `FactoryConfigurationError` with no detail mesage.
FactoryConfiguration-Error(Exception)	Create a new `FactoryConfigurationError` with a given `Exception` base cause of the error.
FactoryConfiguration-Error(Exception, String)	Create a new `FactoryConfigurationError` with the given `Exception` base cause and detail message.
FactoryConfiguration-Error(String)	Create a new `FactoryConfigurationError` with the `String` specified as an error message.

Methods

getException()	Return the actual exception (if any) that caused this exception to be raised.
getMessage()	Return the message (if any) for this error .

Inherited Member Summary

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, printStackTrace, printStackTrace, printStack-Trace, toString`

Methods inherited from class java.lang.Object

Inherited Member Summary
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

FactoryConfigurationError()

```
public FactoryConfigurationError()
```

Create a new `FactoryConfigurationError` with no detail mesage.

FactoryConfigurationError(Exception)

```
public FactoryConfigurationError(java.lang.Exception e)
```

Create a new `FactoryConfigurationError` with a given `Exception` base cause of the error.

Parameters:

`e` - The exception to be encapsulated in a FactoryConfigurationError.

FactoryConfigurationError(Exception, String)

```
public FactoryConfigurationError(java.lang.Exception e, java.lang.String msg)
```

Create a new `FactoryConfigurationError` with the given `Exception` base cause and detail message.

Parameters:

`e` - The exception to be encapsulated in a FactoryConfigurationError

`msg` - The detail message.

`e` - The exception to be wrapped in a FactoryConfigurationError

FactoryConfigurationError(String)

```
public FactoryConfigurationError(java.lang.String msg)
```

Create a new `FactoryConfigurationError` with the `String` specified as an error message.

Parameters:

`msg` - The error message for the exception.

Methods

getException()

```
public java.lang.Exception getException()
```

Return the actual exception (if any) that caused this exception to be raised.

Returns: The encapsulated exception, or null if there is none.

---

getMessage()

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

Return the message (if any) for this error . If there is no message for the exception and there is an encapsulated exception then the message of that exception will be returned.

Overrides: java.lang.Throwable.getMessage() in class java.lang.Throwable

Returns: The error message.

javax.xml.parsers
ParserConfigurationException

Syntax
```
public class ParserConfigurationException extends java.lang.Exception
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.xml.parsers.ParserConfigurationException
```

All Implemented Interfaces: java.io.Serializable

Description
Indicates a serious configuration error.

Since: JAXP 1.0

Member Summary	
Constructors	
ParserConfiguration-Exception()	Create a new `ParserConfigurationException` with no detail mesage.
ParserConfiguration-Exception(String)	Create a new `ParserConfigurationException` with the `String` specified as an error message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`
Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructors

ParserConfigurationException()

ParserConfigurationException(String)

```
public ParserConfigurationException()
```

Create a new `ParserConfigurationException` with no detail mesage.

ParserConfigurationException(String)

```
public ParserConfigurationException(java.lang.String msg)
```

Create a new `ParserConfigurationException` with the `String` specified as an error message.

Parameters:

`msg` - The error message for the exception.

javax.xml.parsers
SAXParser

Syntax

```
public abstract class SAXParser

java.lang.Object
  |
  +--javax.xml.parsers.SAXParser
```

Description

Defines the API that wraps an `org.xml.sax.XMLReader` implementation class. In JAXP 1.0, this class wrapped the `org.xml.sax.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.SAX-ParserFactory". 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` are called.

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

Since: JAXP 1.0

Member Summary

Constructors

`SAXParser()`	

Methods

`getParser()`	Returns the SAX parser that is encapsultated by the implementation of this class.
`getProperty(String)`	returns the particular property requested for in the underlying implementation of org.xml.sax.XMLReader.
`getXMLReader()`	Returns the XMLReader that is encapsulated by the implementation of this class.
`isNamespaceAware()`	Indicates whether or not this parser is configured to understand namespaces.
`isValidating()`	Indicates whether or not this parser is configured to validate XML documents.
`parse(File, DefaultH-andler)`	Parse the content of the file specified as XML using the specified `org.xml.sax.helpers.DefaultHandler`.
`parse(File, Handler-Base)`	Parse the content of the file specified as XML using the specified `org.xml.sax.HandlerBase`.
`parse(InputSource, DefaultHandler)`	Parse the content given `org.xml.sax.InputSource` as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

Member Summary	
parse(InputSource, HandlerBase)	Parse the content given `org.xml.sax.InputSource` as XML using the specified `org.xml.sax.HandlerBase`.
parse(InputStream, DefaultHandler)	Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.helpers.DefaultHandler`.
parse(InputStream, DefaultHandler, String)	Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.helpers.DefaultHandler`.
parse(InputStream, HandlerBase)	Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.HandlerBase`.
parse(InputStream, HandlerBase, String)	Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.HandlerBase`.
parse(String, DefaultHandler)	Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified `org.xml.sax.helpers.DefaultHandler`.
parse(String, HandlerBase)	Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified `org.xml.sax.HandlerBase`.
setProperty(String, Object)	Sets the particular property in the underlying implementation of org.xml.sax.XMLReader.

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

Constructors

SAXParser()

```
protected SAXParser()
```

Methods

getParser()

```
public abstract org.xml.sax.Parser getParser()
```

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

Throws: `SAXException`

getProperty(String)

```
public abstract java.lang.Object getProperty(java.lang.String name)
```

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: `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.

`SAXNotSupportedException`, `SAXNotRecognizedException`

See Also: `org.xml.sax.XMLReader#getProperty`

---

getXMLReader()

```
public abstract org.xml.sax.XMLReader getXMLReader()
```

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

Throws: `SAXException`

---

isNamespaceAware()

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

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

---

isValidating()

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

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

---

parse(File, DefaultHandler)

```
public void parse(java.io.File f, org.xml.sax.helpers.DefaultHandler dh)
```

Parse the content of the file specified as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

Parameters:
`f` - The file containing the XML to parse

`dh` - The SAX Handler to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the File object is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(File, HandlerBase)

parse(InputSource, DefaultHandler)

```
public void parse(java.io.File f, org.xml.sax.HandlerBase hb)
```

Parse the content of the file specified as XML using the specified `org.xml.sax.HandlerBase`.

Parameters:

`f` - The file containing the XML to parse

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the File object is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputSource, DefaultHandler)

```
public void parse(org.xml.sax.InputSource is, org.xml.sax.helpers.DefaultHandler dh)
```

Parse the content given `org.xml.sax.InputSource` as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

Parameters:

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

`dh` - The SAX DefaultHandler to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the InputSource is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputSource, HandlerBase)

```
public void parse(org.xml.sax.InputSource is, org.xml.sax.HandlerBase hb)
```

Parse the content given `org.xml.sax.InputSource` as XML using the specified `org.xml.sax.HandlerBase`.

Parameters:

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

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the InputSource is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputStream, DefaultHandler)

```
public void parse(java.io.InputStream is, org.xml.sax.helpers.DefaultHandler dh)
```

Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

Parameters:

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

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the given InputStream is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputStream, DefaultHandler, String)

```
public void parse(java.io.InputStream is, org.xml.sax.helpers.DefaultHandler dh,
            java.lang.String systemId)
```

Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

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: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the given InputStream is null.

`SAXException`

See Also: `version of this method instead.`

---

parse(InputStream, HandlerBase)

```
public void parse(java.io.InputStream is, org.xml.sax.HandlerBase hb)
```

Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.HandlerBase`.

Parameters:

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

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the given InputStream is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(InputStream, HandlerBase, String)

```
public void parse(java.io.InputStream is, org.xml.sax.HandlerBase hb,
            java.lang.String systemId)
```

Parse the content of the given `java.io.InputStream` instance as XML using the specified `org.xml.sax.HandlerBase`.

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: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the given InputStream is null.

`SAXException`

See Also: `version of this method instead.`

---

parse(String, DefaultHandler)

```
public void parse(java.lang.String uri, org.xml.sax.helpers.DefaultHandler dh)
```

Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified `org.xml.sax.helpers.DefaultHandler`.

Parameters:

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

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the uri is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

parse(String, HandlerBase)

```
public void parse(java.lang.String uri, org.xml.sax.HandlerBase hb)
```

Parse the content described by the giving Uniform Resource Identifier (URI) as XML using the specified `org.xml.sax.HandlerBase`.

Parameters:

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

`hb` - The SAX HandlerBase to use.

Throws: `IOException` - If any IO errors occur.

`IllegalArgumentException` - If the uri is null.

`SAXException`

See Also: `org.xml.sax.DocumentHandler`

---

setProperty(String, Object)

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

Sets the particular property in the underlying implementation of org.xml.sax.XMLReader. A list of the core features and properties can be found at http://www.megginson.com/SAX/Java/features.html

Parameters:

`name` - The name of the property to be set.

`value` - The value of the property to be set.

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.

`SAXNotSupportedException`, `SAXNotRecognizedException`

See Also: `org.xml.sax.XMLReader#setProperty`

javax.xml.parsers
SAXParserFactory

Syntax

```
public abstract class SAXParserFactory
```

```
java.lang.Object
  |
  +--javax.xml.parsers.SAXParserFactory
```

Description

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

Since: JAXP 1.0

Member Summary

Constructors

`SAXParserFactory()`	

Methods

`getFeature(String)`	returns the particular property requested for in the underlying implementation of org.xml.sax.XMLReader.
`isNamespaceAware()`	Indicates whether or not the factory is configured to produce parsers which are namespace aware.
`isValidating()`	Indicates whether or not the factory is configured to produce parsers which validate the XML content during parse.
`newInstance()`	Obtain a new instance of a `SAXParserFactory`.
`newSAXParser()`	Creates a new instance of a SAXParser using the currently configured factory parameters.
`setFeature(String, boolean)`	Sets the particular feature in the underlying implementation of org.xml.sax.XMLReader.
`setNamespaceAware(boolean)`	Specifies that the parser produced by this code will provide support for XML namespaces.
`setValidating(boolean)`	Specifies that the parser produced by this code will validate documents as they are parsed.

Inherited Member Summary

Methods inherited from class java.lang.Object

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

Constructors

SAXParserFactory()

```
protected SAXParserFactory()
```

Methods

getFeature(String)

```
public abstract boolean getFeature(java.lang.String name)
```

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: `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.

`SAXNotSupportedException`, `SAXNotRecognizedException`, `ParserConfigurationException`

See Also: `org.xml.sax.XMLReader#getProperty`

isNamespaceAware()

```
public boolean isNamespaceAware()
```

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

isValidating()

```
public boolean isValidating()
```

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

newInstance()

```
public static SAXParserFactory newInstance()
```

Obtain a new instance of a `SAXParserFactory`. 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.

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

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

---

newSAXParser()

```
public abstract SAXParser newSAXParser()
```

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

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

`SAXException`

---

setFeature(String, boolean)

```
public abstract void setFeature(java.lang.String name, boolean value)
```

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.megginson.com/SAX/Java/features.html

Parameters:

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

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

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.

`SAXNotSupportedException`, `SAXNotRecognizedException`, `ParserConfigurationException`

See Also: `org.xml.sax.XMLReader#setFeature`

---

setNamespaceAware(boolean)

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

Specifies that the parser produced by this code will provide support for XML namespaces.

---

setValidating(boolean)

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

Specifies that the parser produced by this code will validate documents as they are parsed.

Package
javax.xml.transform

Description

This package defines the generic APIs for processing transformation instructions, and performing a transformation from source to result. These interfaces have no dependencies on SAX or the DOM standard, and try to make as few assumptions as possible about the details of the source and result of a transformation. It achieves this by defining `Source` and `Result` interfaces.

To define concrete classes for the user, TrAX defines specializations of the interfaces found at the TrAX root level. These interfaces are found in `javax.xml.transform.sax` , `javax.xml.transform.dom` , and `javax.xml.transform.stream` .

TrAX allows a concrete `TransformerFactory` object to be created from the static function `newInstance()` . The “javax.xml.transform.TransformerFactory” system property determines which factory implementation to instantiate. This property names a concrete subclass of the TransformerFactory abstract class. If this system property is not defined, a platform default is used.

Specification of Inputs and Outputs

TrAX defines two interface objects called `Source` and `Result` . In order to pass Source and Result objects to the TrAX interfaces, concrete classes must be used. TrAX defines three concrete representations for each of these objects: `StreamSource` and `StreamResult` , `SAXSource` and `SAXResult` , and `DOMSource` and `DOMResult` . Each of these objects defines a FEATURE string (which is i the form of a URL), which can be passed into `getFeature(String)` to see if the given type of Source or Result object is supported. For instance, to test if a DOMSource and a StreamResult is supported, you can apply the following test.

```
    TransformerFactory tfactory = TransformerFactory.newInstance();

    if (tfactory.getFeature(DOMSource.FEATURE) && tfactory.getFeature(StreamResult.FEATU
RE))
    {
       ...
    }
Qualified Name representation
```

Namespaces present something of a problem area when dealing with XML objects. Qualified Names appear in XML markup as prefixed names. But the prefixes themselves do not hold identity. Rather, it is the URIs that they contextually map to that hold the identity. Therefore, when passing a Qualified Name like “xyz:foo” among Java programs, one must provide a means to map “xyz” to a namespace.

One solution has been to create a “QName” object that holds the namespace URI, as well as the prefix and local name, but this is not always an optimal solution, as when, for example, you want to use unique strings as keys in a dictionary object. Not having a string representation also makes it difficult to specify a namespaced identity outside the context of an XML document.

In order to pass namespaced values to transformations, for instance as a set of properties to the Serializer, this specification defines that a String “qname” object parameter be passed as two-part string, the namespace URI enclosed in curly braces ({}), followed by the local name. If the qname has a null URI, then the String object only contains the local name. An application can safely check for a non-null URI by testing to see if the first character of the name is a ’{’ character.

For example, if a URI and local name were obtained from an element defined with <xyz:foo xmlns:xyz=“http://xyz.foo.com/yada/baz.html”/>, then the TrAX Qualified Name would be “{http://xyz.foo.com/yada/baz.html}foo”. Note that the prefix is lost.

Result Tree Serialization

Serialization of the result tree to a stream can be controlled with the `setOutputProperties(Properties)` and the `setOutputProperty(String, String)`

methods. Strings that match the XSLT specification for xsl:output attributes can be referenced from the `OutputKeys` class. Other strings can be specified as well. If the transformer does not recognize an output key, a `java.lang.IllegalArgumentException` is thrown, unless the unless the key name is namespace qualified. Output key names that are qualified by a namespace are ignored or passed on to the serializer mechanism.

If all that is desired is the simple identity transformation of a source to a result, then `TransformerFactory` provides a `newTransformer()` method with no arguments. This method creates a Transformer that effectively copies the source to the result. This method may be used to create a DOM from SAX events or to create an XML or HTML stream from a DOM or SAX events. The TrAX APIs throw three types of specialized exceptions. A `TFactoryConfigurationError` is parallel to the `FactoryConfigurationError` , and is thrown when a configuration problem with the TransformerFactory exists. This error will typically be thrown when the transformation factory class specified with the "javax.xml.transform.TransformerFactory" system property cannot be found or instantiated.

A `TransformerConfigurationException` may be thrown if for any reason a Transformer can not be created. A TransformerConfigurationException may be thrown if there is a syntax error in the transformation instructions, for example when `newTransformer(Source)` is called.

`TransformerException` is a general exception that occurs during the course of a transformation. A transformer exception may wrap another exception, and if any of the `printStackTrace()` methods are called on it, it will produce a list of stack dumps, starting from the most recent. The transformer exception also provides a `SourceLocator` object which indicates where in the source tree or transformation instructions the error occurred. `getMessageAndLocation()` may be called to get an error message with location info, and `getLocationAsString()` may be called to get just the location string.

Transformation warnings and errors are normally first sent to a `ErrorListener` , at which point the implementor may decide to report the error or warning, and may decide to throw an exception for a non-fatal error. The error listener may be set via `setErrorListener(ErrorListener)` for reporting errors that have to do with syntax errors in the transformation instructions, or via `setErrorListener(ErrorListener)` to report errors that occur during the transformation. The error listener on both objects should always be valid and non-null, whether set by the user or a default implementation provided by the processor.

Resolution of URIs within a transformation

TrAX provides a way for URIs referenced from within the stylesheet instructions or within the transformation to be resolved by the calling application. This can be done by creating a class that implements the URIResolver interface, with its one method, `resolve(String, String)` , and use this class to set the URI resolution for the transformation instructions or transformation with `setURIResolver(URIResolver)` or `setURIResolver(URIResolver)` . The URIResolver.resolve method takes two String arguments, the URI found in the stylesheet instructions or built as part of the transformation process, and the base URI in effect when the URI passed as the first argument was encountered. The returned `Source` object must be usable by the transformer, as specified in its implemented features.

Class Summary

Interfaces

`ErrorListener`	To provide customized error handling, implement this interface and use the setErrorListener method to register an instance of the implmentation with the Transformer.
`Result`	An object that implements this interface contains the information needed to build a transformation result tree.

Class Summary

Source	An object that implements this interface contains the information needed to act as source input (XML source or transformation instructions).
SourceLocator	This interface is primarily for the purposes of reporting where an error occurred in the XML source or transformation instructions.
Templates	An object that implements this interface is the runtime representation of processed transformation instructions.
URIResolver	An object that implements this interface that can be called by the processor to turn a URI used in document(), xsl:import, or xsl:include into a Source object.
Classes	
OutputKeys	Provides string constants that can be used to set output properties for a Transformer, or to retrieve output properties from a Transformer or Templates object.
Transformer	An instace of this abstract class can transform a source tree into a result tree.
TransformerFactory	A TransformerFactory instance can be used to create Transformer and Template objects.
Exceptions	
TransformerConfigura-tionException	Indicates a serious configuration error.
TransformerException	This class specifies an exceptional condition that occured during the transformation process.
Errors	
TFactoryConfigura-tionError	Thrown when a problem with configuration with the Transformer Factories exists.

javax.xml.transform
ErrorListener

Syntax
```
public interface ErrorListener
```

Description
To provide customized error handling, implement this interface and use the setErrorListener method to register an instance of the implmentation with the Transformer. The Transformer then reports all errors and warnings through this interface.

If an application does *not* register an ErrorListener, errors are reported to System.err.

For transformation errors, a Transformer must use this interface instead of 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 the Transformer is not required to continue with the transformation after a call to fatalError.

Transformers may use this mechanism to report XML parsing errors as well as transformation errors.

Member Summary	
Methods	
error(TransformerException)	Receive notification of a recoverable error.
fatalError(TransformerException)	Receive notification of a non-recoverable error.
warning(TransformerException)	Receive notification of a warning.

Methods

---

error(TransformerException)

```
public void error(TransformerException exception)
```

Receive notification of a recoverable error.

The transformer 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.

Parameters:
`exception` - The error information encapsulated in a transformer exception.

Throws: `TransformerException` - if the application chooses to discontinue the transformation.

See Also: `TransformerException`

---

fatalError(TransformerException)

```
public void fatalError(TransformerException exception)
```

Receive notification of a non-recoverable error.

The application must assume that the transformation cannot continue after the Transformer has invoked this method, and should continue (if at all) only to collect addition error messages. In fact, Transformers are free to stop reporting events once this method has been invoked.

Parameters:
`exception` - The error information encapsulated in a transformer exception.

Throws: `TransformerException` - if the application chooses to discontinue the transformation.

See Also: `TransformerException`

---

warning(TransformerException)

```
public void warning(TransformerException exception)
```

Receive notification of a warning.

Transformers can use this method to report conditions that are not errors or fatal errors. The default behaviour is to take no action.

After invoking this method, the Transformer must continue with the transformation. It should still be possible for the application to process the document through to the end.

Parameters:
`exception` - The warning information encapsulated in a transformer exception.

Throws: `TransformerException` - if the application chooses to discontinue the transformation.

See Also: `TransformerException`

javax.xml.transform
OutputKeys

Syntax

```
public class OutputKeys
```

```
java.lang.Object
  |
  +--javax.xml.transform.OutputKeys
```

Description

Provides string constants that can be used to set output properties for a Transformer, or to retrieve output properties from a Transformer or Templates object.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

Member Summary	
Fields	
CDATA_SECTION_ELEMENTS	cdata-section-elements = *qnames*.
DOCTYPE_PUBLIC	doctype-public = *string*.
DOCTYPE_SYSTEM	doctype-system = *string*.
ENCODING	encoding = *string*.
INDENT	indent = “yes”	“no”.
MEDIA_TYPE	media-type = *string*.
METHOD	method = “xml”	“html”	“text”	*qname-but-not-ncname*.
OMIT_XML_DECLARATION	omit-xml-declaration = “yes”	“no”.
STANDALONE	standalone = “yes”	“no”.
VERSION	version = *nmtoken*.

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

Fields

CDATA_SECTION_ELEMENTS

```
public static final java.lang.String CDATA_SECTION_ELEMENTS
```

cdata-section-elements = *qnames*.

`cdata-section-elements` specifies a list of the names of elements whose text node children should be output using CDATA sections.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

DOCTYPE_PUBLIC

```
public static final java.lang.String DOCTYPE_PUBLIC
```

doctype-public = *string*.

`doctype-public` specifies the public identifier to be used in the document type declaration.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

DOCTYPE_SYSTEM

```
public static final java.lang.String DOCTYPE_SYSTEM
```

doctype-system = *string*.

`doctype-system` specifies the system identifier to be used in the document type declaration.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

ENCODING

```
public static final java.lang.String ENCODING
```

encoding = *string*.

`encoding` specifies the preferred character encoding that the Transformer should use to encode sequences of characters as sequences of bytes. The value of the attribute should be treated case-insensitively. The value must only contain characters in the range #x21 to #x7E (i.e., printable ASCII characters). The value should either be a `charset` registered with the Internet Assigned Numbers Authority [IANA], [RFC2278] or start with `X-`.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

INDENT

```
public static final java.lang.String INDENT
```

indent = “yes” | “no”.

`indent` specifies whether the Transformer may add additional whitespace when outputting the result tree; the value must be `yes` or `no`.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

MEDIA_TYPE

```
public static final java.lang.String MEDIA_TYPE
```

media-type = *string*.

`media-type` specifies the media type (MIME content type) of the data that results from outputting the result tree. The `charset` parameter should not be specified explicitly; instead, when the top-level media type is `text`, a `charset` parameter should be added according to the character encoding actually used by the output method.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

METHOD

`public static final java.lang.String METHOD`

method = “xml” | “html” | “text” | *qname-but-not-ncname*.

The method attribute identifies the overall method that should be used for outputting the result tree. Other non-namespaced values may be used, such as “xhtml”, but, if accepted, the handling of such values is implementation defined. If any of the method values are not accepted and are not namespace qualified, then `setOutputProperty(String, String)` or `setOutputProperties(Properties)` will throw an `java.lang.IllegalArgumentException` .

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

OMIT_XML_DECLARATION

`public static final java.lang.String OMIT_XML_DECLARATION`

omit-xml-declaration = “yes” | “no”.

`omit-xml-declaration` specifies whether the XSLT processor should output an XML declaration; the value must be `yes` or `no`.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

STANDALONE

`public static final java.lang.String STANDALONE`

standalone = “yes” | “no”.

`standalone` specifies whether the Transformer should output a standalone document declaration; the value must be `yes` or `no`.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

VERSION

`public static final java.lang.String VERSION`

version = *nmtoken*.

`version` specifies the version of the output method.

See Also: `<a href="http://www.w3.org/TR/xslt#output">section 16 of the`

javax.xml.transform
Result

Syntax
```
public interface Result
```

All Known Implementing Classes: SAXResult, DOMResult, StreamResult

Description
An object that implements this interface contains the information needed to build a transformation result tree.

Member Summary	
Fields	
PI_DISABLE_OUTPUT_ESCAPING	The name of the processing instruction that is sent if the result tree disables output escaping.
PI_ENABLE_OUTPUT_ESCAPING	The name of the processing instruction that is sent if the result tree enables output escaping at some point after having received a PI_DISABLE_OUTPUT_ESCAPING processing instruction.
Methods	
getSystemId()	Get the system identifier that was set with setSystemId.
setSystemId(String)	Set the system identifier for this Result.

Fields

PI_DISABLE_OUTPUT_ESCAPING

```
public static final java.lang.String PI_DISABLE_OUTPUT_ESCAPING
```

The name of the processing instruction that is sent if the result tree disables output escaping.

Normally, result tree serialization escapes & and < (and possibly other characters) when outputting text nodes. This ensures that the output is well-formed XML. However, it is sometimes convenient to be able to produce output that is almost, but not quite well-formed XML; for example, the output may include ill-formed sections that will be transformed into well-formed XML by a subsequent non-XML aware process. If a processing instruction is sent with this name, serialization should be output without any escaping.

Result DOM trees may also have PI_DISABLE_OUTPUT_ESCAPING and PI_ENABLE_OUTPUT_ESCAPING inserted into the tree.

See Also: `<a href="http://www.w3.org/TR/xslt#disable-output-escaping">disable-output-escaping in XSLT Specification</a>`

PI_ENABLE_OUTPUT_ESCAPING

```
public static final java.lang.String PI_ENABLE_OUTPUT_ESCAPING
```

The name of the processing instruction that is sent if the result tree enables output escaping at some point after having received a PI_DISABLE_OUTPUT_ESCAPING processing instruction.

See Also: `<a href="http://www.w3.org/TR/xslt#disable-output-escaping">disable-output-escaping in XSLT Specification</a>`

Methods

---

getSystemId()

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

Get the system identifier that was set with setSystemId.

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

---

setSystemId(String)

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

Set the system identifier for this Result.

If the Result is not to be written to a file, the system identifier is optional. The application may still want to provide one, however, for use in error messages and warnings, or to resolve relative output identifiers.

Parameters:

`systemId` - The system identifier as a URI string.

javax.xml.transform
Source

Syntax
```
public interface Source
```

All Known Implementing Classes: SAXSource, DOMSource, StreamSource

Description
An object that implements this interface contains the information needed to act as source input (XML source or transformation instructions).

Member Summary	
Methods	
getSystemId()	Get the system identifier that was set with setSystemId.
setSystemId(String)	Set the system identifier for this Source.

Methods

---

getSystemId()

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

Get the system identifier that was set with setSystemId.

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

---

setSystemId(String)

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

Set the system identifier for this Source.

The system identifier is optional if the source does not get its data from a URL, but it may still be useful to provide one. The application can use a system identifier, for example, to resolve relative URIs and to include in error messages and warnings.

Parameters:

`systemId` - The system identifier as a URL string.

javax.xml.transform
SourceLocator

Syntax
```
public interface SourceLocator
```

All Known Subinterfaces: DOMLocator

Description
This interface is primarily for the purposes of reporting where an error occurred in the XML source or transformation instructions.

Member Summary	
Methods	
getColumnNumber()	Return the column number where the current document event ends.
getLineNumber()	Return the line number where the current document event ends.
getPublicId()	Return the public identifier for the current document event.
getSystemId()	Return the system identifier for the current document event.

Methods

getColumnNumber()

```
public int getColumnNumber()
```

Return the column number where the current document event ends.

Warning: The return value from the method is intended only as an approximation for the sake of error reporting; it is not intended to provide sufficient information to edit the character content of the original XML document.

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

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

See Also: getLineNumber()

getLineNumber()

```
public int getLineNumber()
```

Return the line number where the current document event ends.

Warning: The return value from the method is intended only as an approximation for the sake of error reporting; it is not intended to provide sufficient information to edit the character content of the original XML document.

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

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

See Also: `getColumnNumber()`

---

getPublicId()

```
public 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 that triggered the event appears.

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

See Also: `getSystemId()`

---

getSystemId()

```
public 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 that triggered the event appears.

If the system identifier is a URL, the parser must resolve it fully before passing it to the application.

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

See Also: `getPublicId()`

javax.xml.transform
Templates

Syntax
```
public interface Templates
```

Description
An object that implements this interface is the runtime representation of processed transformation instructions.

Templates must be threadsafe for a given instance over multiple threads running concurrently, and may be used multiple times in a given session.

Member Summary	
Methods	
`getOutputProperties()`	Get the static properties for xsl:output.
`newTransformer()`	Create a new transformation context for this Templates object.

Methods

getOutputProperties()

```
public java.util.Properties getOutputProperties()
```

Get the static properties for xsl:output. The object returned will be a clone of the internal values. Accordingly, it can be mutated without mutating the Templates object, and then handed in to the process method.

For XSLT, Attribute Value Templates attribute values will be returned unexpanded (since there is no context at this point).

Returns: A Properties object, never null.

newTransformer()

```
public Transformer newTransformer()
```

Create a new transformation context for this Templates object.

Returns: A valid non-null instance of a Transformer.

Throws: `TransformerConfigurationException` - if a Transformer can not be created.

javax.xml.transform
TFactoryConfigurationError

Syntax

```
public class TFactoryConfigurationError extends java.lang.Error
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Error
              |
              +--javax.xml.transform.TFactoryConfigurationError
```

All Implemented Interfaces: java.io.Serializable

Description

Thrown when a problem with configuration with the Transformer Factories exists. This error will typically be thrown when the class of a transformation factory specified in the system properties cannot be found or instantiated.

Member Summary

Constructors

TFactoryConfigurationError()	Create a new `TFactoryConfigurationError` with no detail mesage.
TFactoryConfigurationError(Exception)	Create a new `TFactoryConfigurationError` with a given `Exception` base cause of the error.
TFactoryConfigurationError(Exception, String)	Create a new `TFactoryConfigurationError` with the given `Exception` base cause and detail message.
TFactoryConfigurationError(String)	Create a new `TFactoryConfigurationError` with the `String` specified as an error message.

Methods

getException()	Return the actual exception (if any) that caused this exception to be raised.
getMessage()	Return the message (if any) for this error .

Inherited Member Summary

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructors

TFactoryConfigurationError()

```
public TFactoryConfigurationError()
```

Create a new `TFactoryConfigurationError` with no detail mesage.

TFactoryConfigurationError(Exception)

```
public TFactoryConfigurationError(java.lang.Exception e)
```

Create a new `TFactoryConfigurationError` with a given `Exception` base cause of the error.

Parameters:

`e` - The exception to be encapsulated in a TFactoryConfigurationError.

TFactoryConfigurationError(Exception, String)

```
public TFactoryConfigurationError(java.lang.Exception e, java.lang.String msg)
```

Create a new `TFactoryConfigurationError` with the given `Exception` base cause and detail message.

Parameters:

`e` - The exception to be encapsulated in a TFactoryConfigurationError

`msg` - The detail message.

`e` - The exception to be wrapped in a TFactoryConfigurationError

---

TFactoryConfigurationError(String)

```
public TFactoryConfigurationError(java.lang.String msg)
```

Create a new `TFactoryConfigurationError` with the `String` specified as an error message.

Parameters:

`msg` - The error message for the exception.

Methods

---

getException()

```
public java.lang.Exception getException()
```

Return the actual exception (if any) that caused this exception to be raised.

Returns: The encapsulated exception, or null if there is none.

---

getMessage()

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

Return the message (if any) for this error . If there is no message for the exception and there is an encapsulated exception then the message of that exception will be returned.

Overrides: java.lang.Throwable.getMessage() in class java.lang.Throwable

Returns: The error message.

javax.xml.transform
Transformer

Syntax

```
public abstract class Transformer
```

```
java.lang.Object
  |
  +--javax.xml.transform.Transformer
```

Description

An instace of this abstract class can transform a source tree into a result tree.

An instance of this class can be obtained with the `TransformerFactory.newTransformer` method. This instance may then be used to process XML from a variety of sources and write the transformation output to a variety of sinks.

*

An object of this class may not be used in multiple threads running concurrently. Different Transformers may be used concurrently by different threads.

A Transformer may be used multiple times. Parameters and output properties are preserved across transformations.

Member Summary

Constructors

`Transformer()`	Default constructor is protected on purpose.

Methods

`clearParameters()`	Clear all parameters set with setParameter.
`getErrorListener()`	Get the error event handler in effect for the transformation.
`getOutputProperties()`	Get a copy of the output properties for the transformation.
`getOutputProperty(String)`	Get an output property that is in effect for the transformation.
`getParameter(String)`	Get a parameter that was explicitly set with setParameter or setParameters.
`getURIResolver()`	Get an object that will be used to resolve URIs used in document(), etc.
`setErrorListener(ErrorListener)`	Set the error event listener in effect for the transformation.
`setOutputProperties(Properties)`	Set the output properties for the transformation.
`setOutputProperty(String, String)`	Set an output property that will be in effect for the transformation.
`setParameter(String, Object)`	Add a parameter for the transformation.
`setURIResolver(URIResolver)`	Set an object that will be used to resolve URIs used in document().
`transform(Source, Result)`	Process the source tree to the output result.

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

Constructors

Transformer()

```
protected Transformer()
```

Default constructor is protected on purpose.

Methods

clearParameters()

```
public abstract void clearParameters()
```

Clear all parameters set with setParameter.

getErrorListener()

```
public abstract ErrorListener getErrorListener()
```

Get the error event handler in effect for the transformation.

Returns: The current error handler, which should never be null.

getOutputProperties()

```
public abstract java.util.Properties getOutputProperties()
```

Get a copy of the output properties for the transformation.

The properties should contain a set of layered properties. The first “layer” will contain the properties that were set with setOutputProperties and setOutputProperty. Subsequent layers contain the properties set in the stylesheet and the default properties for the transformation type. There is no guarantee on how the layers are ordered after the first layer. Thus, getOutputProperties().getProperty(String key) will obtain any property in effect for the stylesheet, while getOutputProperties().get(String key) will only retrieve properties that were explicitly set with setOutputProperties and setOutputProperty.

Note that mutation of the Properties object returned will not effect the properties that the transformation contains.

See Also: `OutputKeys`, `java.util.Properties`

---

getOutputProperty(String)

```
public abstract java.lang.String getOutputProperty(java.lang.String name)
```

Get an output property that is in effect for the transformation. The property specified may be a property that was set with setOutputProperty, or it may be a property specified in the stylesheet.

Parameters:
`name` - A non-null String that specifies an output property name, which may be namespace qualified.

Returns: The string value of the output property, or null if no property was found.

Throws: `IllegalArgumentException` - If the property is not supported.

See Also: `OutputKeys`

---

getParameter(String)

```
public abstract java.lang.Object getParameter(java.lang.String name)
```

Get a parameter that was explicitly set with setParameter or setParameters.

This method does not return a default parameter value, which cannot be determined until the node context is evaluated during the transformation process.

Returns: A parameter that has been set with setParameter or setParameters.

---

getURIResolver()

```
public abstract URIResolver getURIResolver()
```

Get an object that will be used to resolve URIs used in document(), etc.

Returns: An object that implements the URIResolver interface, or null.

---

setErrorListener(ErrorListener)

```
public abstract void setErrorListener(ErrorListener listener)
```

Set the error event listener in effect for the transformation.

Parameters:
`listener` - The new error listener.

Throws: `IllegalArgumentException` - if listener is null.

---

setOutputProperties(Properties)

```
public abstract void setOutputProperties(java.util.Properties oformat)
```

Set the output properties for the transformation. These properties will override properties set in the Templates with xsl:output.

If argument to this function is null, any properties previously set are removed.

Pass a qualified property key name as a two-part string, the namespace URI enclosed in curly braces ({}), followed by the local name. If the name has a null URL, the String only contain the local name. An application can safely check for a non-null URI by testing to see if the first character of the name is a '{' character.

For example, if a URI and local name were obtained from an element defined with <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/>, then the TrAX name would be "{http://xyz.foo.com/yada/baz.html}foo". Note that no prefix is used.

If a given property is not supported, it will be silently ignored.

Parameters:

`oformat` - A set of output properties that will be used to override any of the same properties in effect for the transformation.

Throws: `IllegalArgumentException` - if any of the argument keys are not recognized and are not namespace qualified.

See Also: `OutputKeys`, `java.util.Properties`

---

setOutputProperty(String, String)

```
public abstract void setOutputProperty(java.lang.String name, java.lang.String value)
```

Set an output property that will be in effect for the transformation.

Pass a qualified property name as a two-part string, the namespace URI enclosed in curly braces ({}), followed by the local name. If the name has a null URL, the String only contain the local name. An application can safely check for a non-null URI by testing to see if the first character of the name is a '{' character.

For example, if a URI and local name were obtained from an element defined with <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/>, then the TrAX name would be "{http://xyz.foo.com/yada/baz.html}foo". Note that no prefix is used.

Parameters:

`name` - A non-null String that specifies an output property name, which may be namespace qualified.

`value` - The non-null string value of the output property.

Throws: `IllegalArgumentException` - If the property is not supported, and is not qualified with a namespace.

See Also: `OutputKeys`

---

setParameter(String, Object)

```
public abstract void setParameter(java.lang.String name, java.lang.Object value)
```

Add a parameter for the transformation.

Pass a qualified name as a two-part string, the namespace URI enclosed in curly braces ({}), followed by the local name. If the name has a null URL, the String only contain the local name. An application can safely check for a non-null URI by testing to see if the first character of the name is a '{' character.

For example, if a URI and local name were obtained from an element defined with <xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/>, then the TrAX name would be "{http://xyz.foo.com/yada/baz.html}foo". Note that no prefix is used.

Parameters:

`name` - The name of the parameter, which may begin with a namespace URI in curly braces ({}).

`value` - The value object. This can be any valid Java object. It is up to the processor to provide the proper object coersion or to simply pass the object on for use in an extension.

---

setURIResolver(URIResolver)

```
public abstract void setURIResolver(URIResolver resolver)
```

Set an object that will be used to resolve URIs used in document().

Parameters:

`resolver` - An object that implements the URIResolver interface, or null.

---

transform(Source, Result)

```
public abstract void transform(Source xmlSource, Result outputTarget)
```

Process the source tree to the output result.

Parameters:

`xmlSource` - The input for the source tree.

`outputTarget` - The output target.

Throws: `TransformerException` - If an unrecoverable error occurs during the course of the transformation.

javax.xml.transform
TransformerConfigurationException

Syntax

```
public class TransformerConfigurationException extends TransformerException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--TransformerException
                    |
                    +--javax.xml.transform.TransformerConfigurationException
```

All Implemented Interfaces: java.io.Serializable

Description

Indicates a serious configuration error.

Member Summary

Constructors

TransformerConfigurationException()	Create a new `TransformerConfigurationException` with no detail mesage.
TransformerConfigurationException(Exception)	Create a new `TransformerConfigurationException` with a given `Exception` base cause of the error.
TransformerConfigurationException(String)	Create a new `TransformerConfigurationException` with the `String` specified as an error message.
TransformerConfigurationException(String, Exception)	Create a new `TransformerConfigurationException` with the given `Exception` base cause and detail message.

Inherited Member Summary

Methods inherited from interface TransformerException

getLocator(), getException(), getCause(), initCause(Throwable), getMessageAndLocation(), getLocationAsString(), printStackTrace(), printStackTrace(PrintStream), printStackTrace(PrintWriter)

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, toString`

Methods inherited from class java.lang.Object

TransformerConfigurationException()

Inherited Member Summary
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

TransformerConfigurationException()

```
public TransformerConfigurationException()
```

Create a new `TransformerConfigurationException` with no detail mesage.

TransformerConfigurationException(Exception)

```
public TransformerConfigurationException(java.lang.Exception e)
```

Create a new `TransformerConfigurationException` with a given `Exception` base cause of the error.

Parameters:

`e` - The exception to be encapsulated in a TransformerConfigurationException.

TransformerConfigurationException(String)

```
public TransformerConfigurationException(java.lang.String msg)
```

Create a new `TransformerConfigurationException` with the `String` specified as an error message.

Parameters:

`msg` - The error message for the exception.

TransformerConfigurationException(String, Exception)

```
public TransformerConfigurationException(java.lang.String msg, java.lang.Exception e)
```

Create a new `TransformerConfigurationException` with the given `Exception` base cause and detail message.

Parameters:

`e` - The exception to be encapsulated in a TransformerConfigurationException

`msg` - The detail message.

`e` - The exception to be wrapped in a TransformerConfigurationException

javax.xml.transform
TransformerException

Syntax

```
public class TransformerException extends java.lang.Exception
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--javax.xml.transform.TransformerException
```

Direct Known Subclasses: TransformerConfigurationException

All Implemented Interfaces: java.io.Serializable

Description

This class specifies an exceptional condition that occured during the transformation process.

Member Summary

Constructors

TransformerException(Exception)	Create a new TransformerException wrapping an existing exception.
TransformerException(String)	Create a new TransformerException.
TransformerException(String, Exception)	Wrap an existing exception in a TransformerException.
TransformerException(String, SourceLocator)	Create a new TransformerException from a message and a Locator.
TransformerException(String, SourceLocator, Exception)	Wrap an existing exception in a TransformerException.

Methods

getCause()	Returns the cause of this throwable or `null` if the cause is nonexistent or unknown.
getException()	This method retrieves an exception that this exception wraps.
getLocationAsString()	Get the location information as a string.
getLocator()	Method getLocator retrieves and instance of a SourceLocator object that specifies where an error occured.
getMessageAndLocation()	Get the error message with location information appended.
initCause(Throwable)	Initializes the *cause* of this throwable to the specified value.
printStackTrace()	Print the the trace of methods from where the error originated.
printStackTrace(PrintStream)	Print the the trace of methods from where the error originated.

Member Summary	
printStack-Trace(PrintWriter)	Print the the trace of methods from where the error originated.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, toString`
Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructors

TransformerException(Exception)

```
public TransformerException(java.lang.Exception e)
```

Create a new TransformerException wrapping an existing exception.

Parameters:

`e` - The exception to be wrapped.

TransformerException(String)

```
public TransformerException(java.lang.String message)
```

Create a new TransformerException.

Parameters:

`message` - The error or warning message.

TransformerException(String, Exception)

```
public TransformerException(java.lang.String message, java.lang.Exception e)
```

Wrap an existing exception in a TransformerException.

This is used for throwing processor exceptions before the processing has started.

Parameters:

`message` - The error or warning message, or null to use the message from the embedded exception.

`e` - Any exception

TransformerException(String, SourceLocator)

```
public TransformerException(java.lang.String message, SourceLocator locator)
```

Create a new TransformerException from a message and a Locator.

This constructor is especially useful when an application is creating its own exception from within a DocumentHandler callback.

Parameters:

`message` - The error or warning message.

`locator` - The locator object for the error or warning.

TransformerException(String, SourceLocator, Exception)

```
public TransformerException(java.lang.String message, SourceLocator locator,
            java.lang.Exception e)
```

Wrap an existing exception in a TransformerException.

Parameters:

`message` - The error or warning message, or null to use the message from the embedded exception.

`locator` - The locator object for the error or warning.

`e` - Any exception

Methods

---

getCause()

```
public java.lang.Throwable getCause()
```

Returns the cause of this throwable or `null` if the cause is nonexistent or unknown. (The cause is the throwable that caused this throwable to get thrown.)

---

getException()

```
public java.lang.Exception getException()
```

This method retrieves an exception that this exception wraps.

Returns: An Exception object, or null.

See Also: `getCause()`

---

getLocationAsString()

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

Get the location information as a string.

Returns: A string with location info, or null if there is no location information.

---

getLocator()

```
public SourceLocator getLocator()
```

Method getLocator retrieves and instance of a SourceLocator object that specifies where an error occured.

Returns: A SourceLocator object, or null if none was specified.

---

getMessageAndLocation()

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

Get the error message with location information appended.

---

initCause(Throwable)

```
public synchronized java.lang.Throwable initCause(java.lang.Throwable cause)
```

Initializes the *cause* of this throwable to the specified value. (The cause is the throwable that caused this throwable to get thrown.)

This method can be called at most once. It is generally called from within the constructor, or immediately after creating the throwable. If this throwable was created with `TransformerException(Exception)` or `TransformerException(String, Exception)` , this method cannot be called even once.

Parameters:

`cause` - the cause (which is saved for later retrieval by the `getCause()` method). (A `null` value is permitted, and indicates that the cause is nonexistent or unknown.)

Returns: a reference to this `Throwable` instance.

Throws: `IllegalArgumentException` - if `cause` is this throwable. (A throwable cannot be its own cause.)

`IllegalStateException` - if this throwable was created with `TransformerException(Exception)` or `TransformerException(String, Exception)` , or this method has already been called on this throwable.

---

printStackTrace()

```
public void printStackTrace()
```

Print the the trace of methods from where the error originated. This will trace all nested exception objects, as well as this object.

Overrides: java.lang.Throwable.printStackTrace() in class java.lang.Throwable

---

printStackTrace(PrintStream)

```
public void printStackTrace(java.io.PrintStream s)
```

Print the the trace of methods from where the error originated. This will trace all nested exception objects, as well as this object.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintStream) in class java.lang.Throwable

Parameters:

`s` - The stream where the dump will be sent to.

printStackTrace(PrintWriter)

```
public void printStackTrace(java.io.PrintWriter s)
```

Print the the trace of methods from where the error originated. This will trace all nested exception objects, as well as this object.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintWriter) in class java.lang.Throwable

Parameters:

`s` - The writer where the dump will be sent to.

javax.xml.transform
TransformerFactory

Syntax

```
public abstract class TransformerFactory

java.lang.Object
  |
  +--javax.xml.transform.TransformerFactory
```

Direct Known Subclasses: SAXTransformerFactory

Description

A TransformerFactory instance can be used to create Transformer and Template objects.

The system property that determines which Factory implementation to create is named "javax.xml.transform.TransformerFactory". This property names a concrete subclass of the TransformerFactory abstract class. If the property is not defined, a platform default is be used.

Member Summary

Constructors	
TransformerFactory()	Default constructor is protected on purpose.
Methods	
getAssociated-Stylesheet(Source, String, String, String)	Get the stylesheet specification(s) associated via the xml-stylesheet processing instruction (see http://www.w3.org/TR/xml-stylesheet/) with the document document specified in the source parameter, and that match the given criteria.
getAttribute(String)	Allows the user to retrieve specific attributes on the underlying implementation.
getErrorListener()	Get the error event handler for the TransformerFactory.
getFeature(String)	Look up the value of a feature.
getURIResolver()	Get the object that is used by default during the transformation to resolve URIs used in document(), xsl:import, or xsl:include.
newInstance()	Obtain a new instance of a `Transform Factory`.
newTemplates(Source)	Process the Source into a Templates object, which is a a compiled representation of the source.
newTransformer()	Create a new Transformer object that performs a copy of the source to the result.
newTrans-former(Source)	Process the Source into a Transformer object.
setAttribute(String, Object)	Allows the user to set specific attributes on the underlying implementation.
setErrorLis-tener(ErrorListener)	Set the error event listener for the TransformerFactory, which is used for the processing of transformation instructions, and not for the transformation itself.
setURIResolver(URIRe-solver)	Set an object that is used by default during the transformation to resolve URIs used in xsl:import, or xsl:include.

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

Constructors

TransformerFactory()

```
protected TransformerFactory()
```

Default constructor is protected on purpose.

Methods

getAssociatedStylesheet(Source, String, String, String)

```
public abstract Source getAssociatedStylesheet(Source source, java.lang.String media,
            java.lang.String title, java.lang.String charset)
```

Get the stylesheet specification(s) associated via the xml-stylesheet processing instruction (see http://www.w3.org/TR/xml-stylesheet/) with the document document specified in the source parameter, and that match the given criteria. Note that it is possible to return several stylesheets, in which case they are applied as if they were a list of imports or cascades.

Parameters:

`source` - The XML source document.

`media` - The media attribute to be matched. May be null, in which case the prefered templates will be used (i.e. alternate = no).

`title` - The value of the title attribute to match. May be null.

`charset` - The value of the charset attribute to match. May be null.

Returns: A Source object suitable for passing to the TransformerFactory.

Throws: `TransformerConfigurationException.`, `TransformerConfigurationException`

getAttribute(String)

```
public abstract java.lang.Object getAttribute(java.lang.String name)
```

Allows the user to retrieve specific attributes on the underlying implementation.

Parameters:

`name` - The name of the attribute.

Returns: value The value of the attribute.

Throws: `IllegalArgumentException` - thrown if the underlying implementation doesn't recognize the attribute.

---

getErrorListener()

`public abstract ErrorListener getErrorListener()`

Get the error event handler for the TransformerFactory.

Returns: The current error handler, which should never be null.

---

getFeature(String)

`public abstract boolean getFeature(java.lang.String name)`

Look up the value of a feature.

The feature name is any absolute URI.

Parameters:
`name` - The feature name, which is an absolute URI.

Returns: The current state of the feature (true or false).

---

getURIResolver()

`public abstract URIResolver getURIResolver()`

Get the object that is used by default during the transformation to resolve URIs used in document(), xsl:import, or xsl:include.

Returns: The URIResolver that was set with setURIResolver.

---

newInstance()

`public static TransformerFactory newInstance()`

Obtain a new instance of a `Transform Factory`. This static method creates a new factory instance based on a system property setting or (if this property is not defined) the platform default.

The system property that determines which Factory implementation to create is named "javax.xml.transform.TransformerFactory". This property names a concrete subclass of the Transformer-Factory abstract class. If hte property is not defined, a platform default is used.

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

Returns: new TransformerFactory instance, never null.

Throws: `TFactoryConfigurationError` - if the implmentation is not available or cannot be instantiated.

---

newTemplates(Source)

`public abstract Templates newTemplates(Source source)`

Process the Source into a Templates object, which is a a compiled representation of the source. This Templates object may then be used concurrently across multiple threads. Creating a Templates object allows the TransformerFactory to do detailed performance optimization of transformation instructions, without penalizing runtime transformation.

Parameters:
`source` - An object that holds a URL, input stream, etc.

Returns: A Templates object capable of being used for transformation purposes, never null.

Throws: `TransformerConfigurationException` - May throw this during the parse when it is constructing the Templates object and fails.

---

newTransformer()

```
public abstract Transformer newTransformer()
```

Create a new Transformer object that performs a copy of the source to the result.

Parameters:
`source` - An object that holds a URI, input stream, etc.

Returns: A Transformer object that may be used to perform a transformation in a single thread, never null.

Throws: `TransformerConfigurationException` - May throw this during the parse when it is constructing the Templates object and fails.

---

newTransformer(Source)

```
public abstract Transformer newTransformer(Source source)
```

Process the Source into a Transformer object. Care must be given not to use this object in multiple threads running concurrently. Different TransformerFactories can be used concurrently by different threads.

Parameters:
`source` - An object that holds a URI, input stream, etc.

Returns: A Transformer object that may be used to perform a transformation in a single thread, never null.

Throws: `TransformerConfigurationException` - May throw this during the parse when it is constructing the Templates object and fails.

---

setAttribute(String, Object)

```
public abstract void setAttribute(java.lang.String name, java.lang.Object value)
```

Allows the user to set specific attributes on the underlying implementation. An attribute in this context is defined to be an option that the implementation provides.

Parameters:
`name` - The name of the attribute.

`value` - The value of the attribute.

Throws: `IllegalArgumentException` - thrown if the underlying implementation doesn't recognize the attribute.

setErrorListener(ErrorListener)

```
public abstract void setErrorListener(ErrorListener listener)
```

Set the error event listener for the TransformerFactory, which is used for the processing of transformation instructions, and not for the transformation itself.

Parameters:
`listener` - The new error listener.

Throws: `IllegalArgumentException` - if listener is null.

setURIResolver(URIResolver)

```
public abstract void setURIResolver(URIResolver resolver)
```

Set an object that is used by default during the transformation to resolve URIs used in xsl:import, or xsl:include.

Parameters:
`resolver` - An object that implements the URIResolver interface, or null.

javax.xml.transform
URIResolver

Syntax
```
public interface URIResolver
```

Description
An object that implements this interface that can be called by the processor to turn a URI used in document(), xsl:import, or xsl:include into a Source object.

Member Summary	
Methods	
resolve(String, String)	Called by the processor when it encounters an xsl:include, xsl:import, or document() function.

Methods

---

resolve(String, String)

```
public Source resolve(java.lang.String href, java.lang.String base)
```

Called by the processor when it encounters an xsl:include, xsl:import, or document() function.

Parameters:

`href` - An href attribute, which may be relative or absolute.

`base` - The base URI in effect when the href attribute was encountered.

Returns: A Source object, or null if the href cannot be resolved, and the processor should try to resolve the URI itself.

Throws: `TransformerException` - if an error occurs when trying to resolve the URI.

resolve(String, String)

Package
javax.xml.transform.dom

Description

This package implements DOM-specific transformation APIs.

The `DOMSource` class allows the client of the TrAX implementation to specify a DOM `org.w3c.dom.Node` as the source of the input tree. The model of how the Transformer deals with the DOM tree in terms of mismatches with the XSLT data model or other data models is beyond the scope of this document. Any of the nodes derived from `org.w3c.dom.Node` are legal input.

The `DOMResult` class allows a`org.w3c.dom.Node` to be specified to which result DOM nodes will be appended. If an output node is not specified, the transformer will use `newDocument()` to create an output `org.w3c.dom.Document` node. If a node is specified, it should be one of the following: `org.w3c.dom.Document`, `org.w3c.dom.Element`, or `org.w3c.dom.DocumentFragment`. Specification of any other node type is implementation dependent and undefined by this API. If the result is a `org.w3c.dom.Document`, the output of the transformation must have a single element root to set as the document element.

The `DOMLocator` node may be passed to `TransformerException` objects, and retrieved by trying to cast the result of the `getLocator()` method. The implementation has no responsibility to use a DOMLocator instead of a `SourceLocator` (though line numbers and the like do not make much sense for a DOM), so the result of getLocator must always be tested with an instanceof.

Class Summary

Interfaces	
`DOMLocator`	Indicates the position of a node in a source DOM, intended primarily for error reporting.
Classes	
`DOMResult`	Acts as a holder for a transformation result tree, in the form of a Document Object Model (DOM) tree.
`DOMSource`	Acts as a holder for a transformation Source tree in the form of a Document Object Model (DOM) tree.

javax.xml.transform.dom
DOMLocator

Syntax
```
public interface DOMLocator extends SourceLocator
```

All Superinterfaces: SourceLocator

Description
Indicates the position of a node in a source DOM, intended primarily for error reporting. To use a SourceLocator, the receiver of an error must downcast the SourceLocator object returned by an exception. A Transformer may use this object for purposes other than error reporting, for instance, to indicate the source node that originated a result node.

Member Summary	
Methods	
getOriginatingNode()	Return the node where the event occurred.

Inherited Member Summary
Methods inherited from interface SourceLocator
getPublicId(), getSystemId(), getLineNumber(), getColumnNumber()

Methods

getOriginatingNode()

```
public org.w3c.dom.Node getOriginatingNode()
```

Return the node where the event occurred.

Returns: The node that is the location for the event.

javax.xml.transform.dom
DOMResult

Syntax

```
public class DOMResult implements Result

java.lang.Object
  |
  +--javax.xml.transform.dom.DOMResult
```

All Implemented Interfaces: Result

Description

Acts as a holder for a transformation result tree, in the form of a Document Object Model (DOM) tree. If no output DOM source is set, the transformation will create a Document node as the holder for the result of the transformation, which may be retrieved wiht getNode.

Member Summary

Fields	
FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Result output of this type.
Constructors	
DOMResult()	Zero-argument default constructor.
DOMResult(Node)	Use a DOM node to create a new output target.
DOMResult(Node, String)	Create a new output target with a DOM node.
Methods	
getNode()	Get the node that will contain the result DOM tree.
getSystemId()	Get the system identifier that was set with setSystemId.
setNode(Node)	Set the node that will contain the result DOM tree.
setSystemId(String)	Method setSystemId Set the systemID that may be used in association with the node.

Inherited Member Summary

Fields inherited from interface Result

PI_DISABLE_OUTPUT_ESCAPING, PI_ENABLE_OUTPUT_ESCAPING

Methods inherited from class java.lang.Object

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

FEATURE

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Result output of this type.

Constructors

DOMResult()

```
public DOMResult()
```

Zero-argument default constructor. In practice, the node should be a `org.w3c.dom.Document` node, a `org.w3c.dom.DocumentFragment` node, or a `org.w3c.dom.Element` node. In other words, a node that accepts children.

DOMResult(Node)

```
public DOMResult(org.w3c.dom.Node node)
```

Use a DOM node to create a new output target. In practice, the node should be a `org.w3c.dom.Document` node, a `org.w3c.dom.DocumentFragment` node, or a `org.w3c.dom.Element` node. In other words, a node that accepts children.

Parameters:

`n` - The DOM node that will contain the result tree.

DOMResult(Node, String)

```
public DOMResult(org.w3c.dom.Node node, java.lang.String systemID)
```

Create a new output target with a DOM node. In practice, the node should be a `org.w3c.dom.Document` node, a `org.w3c.dom.DocumentFragment` node, or a `org.w3c.dom.Element` node. In other words, a node that accepts children.

Parameters:

`node` - The DOM node that will contain the result tree.

`systemID` - The system identifier which may be used in association with this node.

Methods

getNode()

```
public org.w3c.dom.Node getNode()
```

Get the node that will contain the result DOM tree. If no node was set via setNode, the node will be set by the transformation, and may be obtained from this method once the transformation is complete.

Returns: The node to which the transformation will be appended.

---

getSystemId()

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

Get the system identifier that was set with setSystemId.

Specified By: `getSystemId()` in interface `Result`

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

---

setNode(Node)

```
public void setNode(org.w3c.dom.Node node)
```

Set the node that will contain the result DOM tree. In practice, the node should be a `org.w3c.dom.Document` node, a `org.w3c.dom.DocumentFragment` node, or a `org.w3c.dom.Element` node. In other words, a node that accepts children.

Parameters:

`node` - The node to which the transformation will be appended.

---

setSystemId(String)

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

Method setSystemId Set the systemID that may be used in association with the node.

Specified By: `setSystemId(String)` in interface `Result`

Parameters:

`systemId` - The system identifier as a URI string.

javax.xml.transform.dom
DOMSource

Syntax

```
public class DOMSource implements Source

java.lang.Object
  |
  +--javax.xml.transform.dom.DOMSource
```

All Implemented Interfaces: Source

Description

Acts as a holder for a transformation Source tree in the form of a Document Object Model (DOM) tree.

See Also: `<a href="http://www.w3.org/TR/DOM-Level-2">Document Object Model (DOM) Level 2 Specification</a>`

Member Summary

Fields

FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructors

DOMSource()	Zero-argument default constructor.
DOMSource(Node)	Create a new input source with a DOM node.
DOMSource(Node, String)	Create a new input source with a DOM node, and with the system ID also passed in as the base URI.

Methods

getNode()	Get the node that represents a Source DOM tree.
getSystemId()	Get the base ID (URL or system ID) from where URLs will be resolved.
setNode(Node)	Set the node that will represents a Source DOM tree.
setSystemId(String)	Set the base ID (URL or system ID) from where URLs will be resolved.

Inherited Member Summary

Methods inherited from class java.lang.Object

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

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructors

DOMSource()

```
public DOMSource()
```

Zero-argument default constructor. If this is used, and no DOM source is set, then the Transformer will create an empty source Document using `newDocument()` .

DOMSource(Node)

```
public DOMSource(org.w3c.dom.Node n)
```

Create a new input source with a DOM node. The operation will be applied to the subtree rooted at this node. In XSLT, a "/" pattern still means the root of the tree (not the subtree), and the evaluation of global variables and parameters is done from the root node also.

Parameters:

`n` - The DOM node that will contain the Source tree.

DOMSource(Node, String)

```
public DOMSource(org.w3c.dom.Node node, java.lang.String systemID)
```

Create a new input source with a DOM node, and with the system ID also passed in as the base URI.

Parameters:

`node` - The DOM node that will contain the Source tree.

`systemID` - Specifies the base URI associated with node.

Methods

getNode()

```
public org.w3c.dom.Node getNode()
```

Get the node that represents a Source DOM tree.

Returns: The node that is to be transformed.

---

getSystemId()

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

Get the base ID (URL or system ID) from where URLs will be resolved.

Specified By: `getSystemId()` in interface `Source`

Returns: Base URL for this DOM tree.

---

setNode(Node)

```
public void setNode(org.w3c.dom.Node node)
```

Set the node that will represents a Source DOM tree.

Parameters:

`node` - The node that is to be transformed.

---

setSystemId(String)

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

Set the base ID (URL or system ID) from where URLs will be resolved.

Specified By: `setSystemId(String)` in interface `Source`

Parameters:

`baseID` - Base URL for this DOM tree.

Package
javax.xml.transform.sax

Description

This package implements SAX2-specific transformation APIs. It provides classes which allow input from `org.xml.sax.ContentHandler` events, and also classes that produce org.xml.sax.ContentHandler events. It also provides methods to set the input source as an `org.xml.sax.XMLReader`, or to use a `org.xml.sax.InputSource` as the source. It also allows the creation of a `org.xml.sax.XMLFilter`, which enables transformations to "pull" from other transformations, and lets the transformer to be used polymorphically as an `org.xml.sax.XMLReader`.

The `SAXSource` class allows the setting of an `org.xml.sax.XMLReader` to be used for "pulling" parse events, and an `org.xml.sax.InputSource` that may be used to specify the SAX source.

The `SAXResult` class allows the setting of a `org.xml.sax.ContentHandler` to be the receiver of SAX2 events from the transformation. The `SAXTransformerFactory` extends `TransformerFactory` to provide factory methods for creating `TemplatesHandler`, `TransformerHandler`, and `org.xml.sax.XMLReader` instances.

To obtain a `SAXTransformerFactory`, the caller must cast the `TransformerFactory` instance returned from `newInstance()`. The `TransformerHandler` interface allows a transformation to be created from SAX2 parse events, which is a "push" model rather than the "pull" model that normally occurs for a transformation. Normal parse events are received through the `org.xml.sax.ContentHandler` interface, lexical events such as startCDATA and endCDATA are received through the `org.xml.sax.ext.LexicalHandler` interface, and events that signal the start or end of disabling output escaping are received via `org.xml.sax.ContentHandler#processingInstruction`, with the target parameter being `PI_DISABLE_OUTPUT_ESCAPING` and `PI_ENABLE_OUTPUT_ESCAPING`. If parameters, output properties, or other features need to be set on the Transformer handler, a `Transformer` reference will need to be obtained from `getTransformer()`, and the methods invoked from that reference. The `TemplatesHandler` interface allows the creation of `Templates` objects from SAX2 parse events. Once the `org.xml.sax.ContentHandler` events are complete, the Templates object may be obtained from `getTemplates()`. Note that `setSystemId(String)` should normally be called in order to establish a base system ID from which relative URLs may be resolved. The `newXMLFilter(Source)` method allows the creation of a `org.xml.sax.XMLFilter`, which encapsulates the SAX2 notion of a "pull" transformation.

Class Summary

Interfaces

`TemplatesHandler`	A SAX ContentHandler that may be used to process SAX parse events (parsing transformation instructions) into a Templates object.
`TransformerHandler`	The TransformerHandler provides a reference to an object that implements this interface, and that can listen to SAX ContentHandler parse events and transform them to a Result.

Classes

`SAXResult`	Acts as an holder for a transformation Result.
`SAXSource`	Acts as an holder for SAX-style Source.
`SAXTransformerFactory`	This class extends TransformerFactory to provide SAX-specific factory methods.

javax.xml.transform.sax
SAXResult

Syntax

```
public class SAXResult implements Result

java.lang.Object
  |
  +--javax.xml.transform.sax.SAXResult
```

All Implemented Interfaces: Result

Description

Acts as an holder for a transformation Result.

Member Summary

Fields	
FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Result output of this type.
Constructors	
SAXResult()	Zero-argument default constructor.
SAXResult(ContentHandler)	Create a SAXResult that targets a SAX2 ContentHandler.
Methods	
getHandler()	Get the ContentHandler that is the Result.
getSystemId()	Get the system identifier that was set with setSystemId.
setHandler(ContentHandler)	Set the target to be a SAX2 ContentHandler.
setSystemId(String)	Method setSystemId Set the systemID that may be used in association with the ContentHandler.

Inherited Member Summary

Fields inherited from interface Result

PI_DISABLE_OUTPUT_ESCAPING, PI_ENABLE_OUTPUT_ESCAPING

Methods inherited from class java.lang.Object

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

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Result output of this type.

Constructors

SAXResult()

```
public SAXResult()
```

Zero-argument default constructor.

SAXResult(ContentHandler)

```
public SAXResult(org.xml.sax.ContentHandler handler)
```

Create a SAXResult that targets a SAX2 ContentHandler.

Parameters:
`handler` - Must be a non-null ContentHandler reference.

Methods

getHandler()

```
public org.xml.sax.ContentHandler getHandler()
```

Get the ContentHandler that is the Result.

Returns: The ContentHandler that is to be transformation output.

getSystemId()

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

Get the system identifier that was set with setSystemId.

Specified By: `getSystemId()` in interface `Result`

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

setHandler(ContentHandler)

```
public void setHandler(org.xml.sax.ContentHandler handler)
```

Set the target to be a SAX2 ContentHandler.

Parameters:

`handler` - Must be a non-null ContentHandler reference.

---

setSystemId(String)

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

Method setSystemId Set the systemID that may be used in association with the ContentHandler.

Specified By: `setSystemId(String)` in interface `Result`

Parameters:

`systemId` - The system identifier as a URI string.

javax.xml.transform.sax
SAXSource

Syntax

```
public class SAXSource implements Source

java.lang.Object
  |
  +--javax.xml.transform.sax.SAXSource
```

All Implemented Interfaces: Source

Description

Acts as an holder for SAX-style Source.

Member Summary

Fields	
FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Source input of this type.
Constructors	
SAXSource()	Zero-argument default constructor.
SAXSource(InputSource)	Create a SAXSource, using a SAX InputSource.
SAXSource(XMLReader, InputSource)	Create a SAXSource, using an XMLReader and a SAX InputSource.
Methods	
getInputSource()	Get the SAX InputSource to be used for the Source.
getSystemId()	Get the base ID (URI or system ID) from where URIs will be resolved.
getXMLReader()	Get the XMLReader to be used for the Source.
setInputSource(InputSource)	Set the SAX InputSource to be used for the Source.
setSystemId(String)	Set the system identifier for this Source.
setXMLReader(XMLReader)	Set the XMLReader to be used for the Source.
sourceToInputSource(Source)	Attempt to obtain a SAX InputSource object from a TrAX Source object.

Inherited Member Summary

Methods inherited from class java.lang.Object

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

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructors

SAXSource()

```
public SAXSource()
```

Zero-argument default constructor. If this constructor is used, and no other method is called, the Transformer assumes an empty input tree, with a default root node.

SAXSource(InputSource)

```
public SAXSource(org.xml.sax.InputSource inputSource)
```

Create a SAXSource, using a SAX InputSource. The Transformer or SAXTransformerFactory creates a reader via org.xml.sax.helpers.XMLReaderFactory (if setXMLReader is not used), sets itself as the reader's ContentHandler, and calls reader.parse(inputSource).

Parameters:

`inputSource` - An input source reference that must be non-null and that will be passed to the parse method of the reader.

SAXSource(XMLReader, InputSource)

```
public SAXSource(org.xml.sax.XMLReader reader, org.xml.sax.InputSource inputSource)
```

Create a SAXSource, using an XMLReader and a SAX InputSource. The Transformer or SAXTransformerFactory will set itself to be the reader's ContentHandler, and then will call reader.parse(inputSource).

Parameters:

`reader` - An XMLReader to be used for the parse.

`inputSource` - A SAX input source reference that must be non-null and that will be passed to the reader parse method.

Methods

getInputSource()

```
public org.xml.sax.InputSource getInputSource()
```

Get the SAX InputSource to be used for the Source.

Returns: A valid InputSource reference, or null.

---

getSystemId()

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

Get the base ID (URI or system ID) from where URIs will be resolved.

Specified By: `getSystemId()` in interface `Source`

Returns: Base URL for the Source, or null.

---

getXMLReader()

```
public org.xml.sax.XMLReader getXMLReader()
```

Get the XMLReader to be used for the Source.

Returns: A valid XMLReader or XMLFilter reference, or null.

---

setInputSource(InputSource)

```
public void setInputSource(org.xml.sax.InputSource inputSource)
```

Set the SAX InputSource to be used for the Source.

Parameters:
`inputSource` - A valid InputSource reference.

---

setSystemId(String)

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

Set the system identifier for this Source. If an input source has already been set, it will set the system ID or that input source, otherwise it will create a new 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 no byte stream or character stream is specified).

Specified By: `setSystemId(String)` in interface `Source`

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

---

setXMLReader(XMLReader)

```
public void setXMLReader(org.xml.sax.XMLReader reader)
```

Set the XMLReader to be used for the Source.

Parameters:
`reader` - A valid XMLReader or XMLFilter reference.

sourceToInputSource(Source)

```
public static org.xml.sax.InputSource sourceToInputSource(Source source)
```

Attempt to obtain a SAX InputSource object from a TrAX Source object.

Parameters:

`source` - Must be a non-null Source reference.

Returns: An InputSource, or null if Source can not be converted.

javax.xml.transform.sax
SAXTransformerFactory

Syntax

```
public abstract class SAXTransformerFactory extends TransformerFactory

java.lang.Object
  |
  +--TransformerFactory
        |
        +--javax.xml.transform.sax.SAXTransformerFactory
```

Description

This class extends TransformerFactory to provide SAX-specific factory methods. It provides two types of ContentHandlers, one for creating Transformers, the other for creating Templates objects.

If an application wants to set the ErrorHandler or EntityResolver for an XMLReader used during a transformation, it should use a URIResolver to return the SAXSource which provides (with getXMLReader) a reference to the XMLReader.

Member Summary

Fields

FEATURE	If `getFeature(String)` returns true when passed this value as an argument, the TransformerFactory returned from `newInstance()` may be safely cast to a SAXTransformerFactory.
FEATURE_XMLFILTER	If `getFeature(String)` returns true when passed this value as an argument, the `newXMLFilter(Source)` and `newXMLFilter(Templates)` methods are supported.

Constructors

SAXTransformerFactory()	The default constructor is protected on purpose.

Methods

newTemplatesHandler()	Get a TemplatesHandler object that can process SAX ContentHandler events into a Templates object.
newTransformerHandler()	Get a TransformerHandler object that can process SAX ContentHandler events into a Result.
newTransformerHandler(Source)	Get a TransformerHandler object that can process SAX ContentHandler events into a Result, based on the transformation instructions specified by the argument.
newTransformerHandler(Templates)	Get a TransformerHandler object that can process SAX ContentHandler events into a Result, based on the Templates argument.
newXMLFilter(Source)	Create an XMLFilter that uses the given Source as the transformation instructions.
newXMLFilter(Templates)	Create an XMLFilter, based on the Templates argument..

Inherited Member Summary
Methods inherited from class `TransformerFactory` `newInstance()`, `newTransformer(Source)`, `newTransformer()`, `newTemplates(Source)`, `getAssociatedStylesheet(Source, String, String, String)`, `setURIResolver(URIResolver)`, `getURIResolver()`, `getFeature(String)`, `setAttribute(String, Object)`, `getAttribute(String)`, `setErrorListener(ErrorListener)`, `getErrorListener()`
Methods inherited from class java.lang.Object `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the TransformerFactory returned from `newInstance()` may be safely cast to a SAXTransformerFactory.

FEATURE_XMLFILTER

```
public static final java.lang.String FEATURE_XMLFILTER
```

If `getFeature(String)` returns true when passed this value as an argument, the `newXMLFilter(Source)` and `newXMLFilter(Templates)` methods are supported.

Constructors

SAXTransformerFactory()

```
protected SAXTransformerFactory()
```

The default constructor is protected on purpose.

Methods

newTemplatesHandler()

```
public abstract TemplatesHandler newTemplatesHandler()
```

Get a TemplatesHandler object that can process SAX ContentHandler events into a Templates object.

Returns: A non-null reference to a TransformerHandler, that may be used as a ContentHandler for SAX parse events.

Throws: `TransformerConfigurationException` - If for some reason the TemplatesHandler cannot be created.

---

newTransformerHandler()

```
public abstract TransformerHandler newTransformerHandler()
```

Get a TransformerHandler object that can process SAX ContentHandler events into a Result. The transformation is defined as an identity (or copy) transformation, for example to copy a series of SAX parse events into a DOM tree.

Returns: A non-null reference to a TransformerHandler, that may be used as a ContentHandler for SAX parse events.

Throws: `TransformerConfigurationException` - If for some reason the TransformerHandler cannot be created.

---

newTransformerHandler(Source)

```
public abstract TransformerHandler newTransformerHandler(Source src)
```

Get a TransformerHandler object that can process SAX ContentHandler events into a Result, based on the transformation instructions specified by the argument.

Parameters:
`src` - The Source of the transformation instructions.

Returns: TransformerHandler ready to transform SAX events.

Throws: `TransformerConfigurationException` - If for some reason the TransformerHandler can not be created.

---

newTransformerHandler(Templates)

```
public abstract TransformerHandler newTransformerHandler(Templates templates)
```

Get a TransformerHandler object that can process SAX ContentHandler events into a Result, based on the Templates argument.

Parameters:
`templates` - The compiled transformation instructions.

Returns: TransformerHandler ready to transform SAX events.

Throws: `TransformerConfigurationException` - If for some reason the TransformerHandler can not be created.

---

newXMLFilter(Source)

```
public abstract org.xml.sax.XMLFilter newXMLFilter(Source src)
```

Create an XMLFilter that uses the given Source as the transformation instructions.

Parameters:

`src` - The Source of the transformation instructions.

Returns: An XMLFilter object, or null if this feature is not supported.

Throws: `TransformerConfigurationException` - If for some reason the TemplatesHandler cannot be created.

---

newXMLFilter(Templates)

```
public abstract org.xml.sax.XMLFilter newXMLFilter(Templates templates)
```

Create an XMLFilter, based on the Templates argument..

Parameters:
`templates` - The compiled transformation instructions.

Returns: An XMLFilter object, or null if this feature is not supported.

Throws: `TransformerConfigurationException` - If for some reason the TemplatesHandler cannot be created.

javax.xml.transform.sax
TemplatesHandler

Syntax
```
public interface TemplatesHandler
```

Description
A SAX ContentHandler that may be used to process SAX parse events (parsing transformation instructions) into a Templates object.

Note that TemplatesHandler does not need to implement LexicalHandler.

Member Summary	
Methods	
getSystemId()	Get the base ID (URI or system ID) from where relative URLs will be resolved.
getTemplates()	When a TemplatesHandler object is used as a ContentHandler or DocumentHandler for the parsing of transformation instructions, it creates a Templates object, which the caller can get once the SAX events have been completed.
setSystemId(String)	Set the base ID (URI or system ID) for the Templates object created by this builder.

Methods

getSystemId()

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

Get the base ID (URI or system ID) from where relative URLs will be resolved.

Returns: The systemID that was set with setSystemId(String) .

getTemplates()

```
public Templates getTemplates()
```

When a TemplatesHandler object is used as a ContentHandler or DocumentHandler for the parsing of transformation instructions, it creates a Templates object, which the caller can get once the SAX events have been completed.

Returns: The Templates object that was created during the SAX event process, or null if no Templates object has been created.

setSystemId(String)

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

setSystemId(String)

Set the base ID (URI or system ID) for the Templates object created by this builder. This must be set in order to resolve relative URIs in the stylesheet. This must be called before the startDocument event.

Parameters:

`baseID` - Base URI for this stylesheet.

javax.xml.transform.sax
TransformerHandler

Syntax
```
public interface TransformerHandler
```

Description
The TransformerHandler provides a reference to an object that implements this interface, and that can listen to SAX ContentHandler parse events and transform them to a Result.

Member Summary	
Methods	
getSystemId()	Get the base ID (URI or system ID) from where relative URLs will be resolved.
getTransformer()	Get the Transformer associated with this handler, which is needed in order to set parameters and output properties.
setResult(Result)	Enables the user of the TransformerHandler to set the to set the Result for the transformation.
setSystemId(String)	Set the base ID (URI or system ID) from where relative URLs will be resolved.

Methods

---

getSystemId()

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

Get the base ID (URI or system ID) from where relative URLs will be resolved.

Returns: The systemID that was set with setSystemId(String) .

---

getTransformer()

```
public Transformer getTransformer()
```

Get the Transformer associated with this handler, which is needed in order to set parameters and output properties.

---

setResult(Result)

```
public void setResult(Result result)
```

Enables the user of the TransformerHandler to set the to set the Result for the transformation.

Parameters:
`result` - A Result instance, should not be null.

Throws: `IllegalArgumentException` - if result is invalid for some reason.

---

setSystemId(String)

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

Set the base ID (URI or system ID) from where relative URLs will be resolved.

Parameters:

`systemID` - Base URI for the source tree.

Package
javax.xml.transform.stream

Description

This package implements stream- and URI- specific transformation APIs.

The `StreamSource` class provides methods for specifying `java.io.InputStream` input, `java.io.Reader` input, and URL input in the form of strings. Even if an input stream or reader is specified as the source, `setSystemId(String)` should still be called, so that the transformer can know from where it should resolve relative URIs. The public identifier is always optional: if the application writer includes one, it will be provided as part of the `SourceLocator` information.

The `StreamResult` class provides methods for specifying `java.io.OutputStream`, `java.io.Writer`, or an output system ID, as the output of the transformation result.

Normally streams should be used rather than readers or writers, for both the Source and Result, since readers and writers already have the encoding established to and from the internal Unicode format. However, there are times when it is useful to write to a character stream, such as when using a StringWriter in order to write to a String, or in the case of reading source XML from a StringReader.

Class Summary	
Classes	
`StreamResult`	Acts as an holder for a transformation result, which may be XML, plain Text, HTML, or some other form of markup.
`StreamSource`	Acts as an holder for a transformation Source in the form of a stream of XML markup.

javax.xml.transform.stream
StreamResult

Syntax

```
public class StreamResult implements Result

java.lang.Object
  |
  +--javax.xml.transform.stream.StreamResult
```

All Implemented Interfaces: Result

Description

Acts as an holder for a transformation result, which may be XML, plain Text, HTML, or some other form of markup.

Member Summary

Fields

FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Result output of this type.

Constructors

StreamResult()	Zero-argument default constructor.
StreamResult(File)	Construct a StreamResult from a File.
StreamResult(OutputStream)	Construct a StreamResult from a byte stream.
StreamResult(String)	Construct a StreamResult from a URL.
StreamResult(Writer)	Construct a StreamResult from a character stream.

Methods

getOutputStream()	Get the byte stream that was set with setOutputStream.
getSystemId()	Get the system identifier that was set with setSystemId.
getWriter()	Get the character stream that was set with setWriter.
setOutputStream(OutputStream)	Set the ByteStream that is to be written to.
setSystemId(File)	Set the system ID from a File reference.
setSystemId(String)	Set the systemID that may be used in association with the byte or character stream, or, if neither is set, use this value as a writeable URI (probably a file name).
setWriter(Writer)	Set the writer that is to receive the result.

Inherited Member Summary

Fields inherited from interface Result

PI_DISABLE_OUTPUT_ESCAPING, PI_ENABLE_OUTPUT_ESCAPING

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

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Result output of this type.

Constructors

StreamResult()

```
public StreamResult()
```

Zero-argument default constructor.

StreamResult(File)

```
public StreamResult(java.io.File f)
```

Construct a StreamResult from a File.

Parameters:
`f` - Must a non-null File reference.

StreamResult(OutputStream)

```
public StreamResult(java.io.OutputStream outputStream)
```

Construct a StreamResult from a byte stream. Normally, a stream should be used rather than a reader, so that the transformer may use instructions contained in the transformation instructions to control the encoding.

Parameters:
`outputStream` - A valid OutputStream reference.

StreamResult(String)

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

StreamResult(Writer)

Construct a StreamResult from a URL.

Parameters:
`systemId` - Must be a String that conforms to the URI syntax.

StreamResult(Writer)

```
public StreamResult(java.io.Writer writer)
```

Construct a StreamResult from a character stream. Normally, a stream should be used rather than a reader, so that the transformer may use instructions contained in the transformation instructions to control the encoding. However, there are times when it is useful to write to a character stream, such as when using a StringWriter.

Parameters:
`writer` - A valid Writer reference.

Methods

---

getOutputStream()

```
public java.io.OutputStream getOutputStream()
```

Get the byte stream that was set with setOutputStream.

Returns: The byte stream that was set with setOutputStream, or null if setOutputStream or the ByteStream constructor was not called.

---

getSystemId()

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

Get the system identifier that was set with setSystemId.

Specified By: `getSystemId()` in interface `Result`

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

---

getWriter()

```
public java.io.Writer getWriter()
```

Get the character stream that was set with setWriter.

Returns: The character stream that was set with setWriter, or null if setWriter or the Writer constructor was not called.

---

setOutputStream(OutputStream)

```
public void setOutputStream(java.io.OutputStream outputStream)
```

Set the ByteStream that is to be written to. Normally, a stream should be used rather than a reader, so that the transformer may use instructions contained in the transformation instructions to control the encoding.

Parameters:
`outputStream` - A valid OutputStream reference.

---

setSystemId(File)

```
public void setSystemId(java.io.File f)
```

Set the system ID from a File reference.

Parameters:
`f` - Must a non-null File reference.

---

setSystemId(String)

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

Set the systemID that may be used in association with the byte or character stream, or, if neither is set, use this value as a writeable URI (probably a file name).

Specified By: `setSystemId(String)` in interface `Result`

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

---

setWriter(Writer)

```
public void setWriter(java.io.Writer writer)
```

Set the writer that is to receive the result. Normally, a stream should be used rather than a writer, so that the transformer may use instructions contained in the transformation instructions to control the encoding. However, there are times when it is useful to write to a writer, such as when using a StringWriter.

Parameters:
`writer` - A valid Writer reference.

javax.xml.transform.stream
StreamSource

Syntax

```
public class StreamSource implements Source

java.lang.Object
  |
  +--javax.xml.transform.stream.StreamSource
```

All Implemented Interfaces: Source

Description

Acts as an holder for a transformation Source in the form of a stream of XML markup.

Member Summary

Fields

FEATURE	If getFeature(String) returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructors

StreamSource()	Zero-argument default constructor.
StreamSource(File)	Construct a StreamSource from a File.
StreamSource(InputStream)	Construct a StreamSource from a byte stream.
StreamSource(InputStream, String)	Construct a StreamSource from a byte stream.
StreamSource(Reader)	Construct a StreamSource from a character reader.
StreamSource(Reader, String)	Construct a StreamSource from a character reader.
StreamSource(String)	Construct a StreamSource from a URL.

Methods

getInputStream()	Get the byte stream that was set with setByteStream.
getPublicId()	Get the public identifier that was set with setPublicId.
getReader()	Get the character stream that was set with setReader.
getSystemId()	Get the system identifier that was set with setSystemId.
setInputStream(InputStream)	Set the byte stream to be used as input.
setPublicId(String)	Set the public identifier for this Source.
setReader(Reader)	Set the input to be a character reader.
setSystemId(File)	Set the system ID from a File reference.
setSystemId(String)	Set the system identifier for this Source.

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

Fields

FEATURE

```
public static final java.lang.String FEATURE
```

If `getFeature(String)` returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructors

StreamSource()

```
public StreamSource()
```

Zero-argument default constructor. If this constructor is used, and no other method is called, the transformer will assume an empty input tree, with a default root node.

StreamSource(File)

```
public StreamSource(java.io.File f)
```

Construct a StreamSource from a File.

Parameters:
`f` - Must a non-null File reference.

StreamSource(InputStream)

```
public StreamSource(java.io.InputStream inputStream)
```

Construct a StreamSource from a byte stream. Normally, a stream should be used rather than a reader, so the XML parser can resolve character encoding specified by the XML declaration.

If this constructor is used to process a stylesheet, normally setSystemId should also be called, so that relative URI references can be resolved.

Parameters:
`inputStream` - A valid InputStream reference to an XML stream.

StreamSource(InputStream, String)

StreamSource(Reader)

```
public StreamSource(java.io.InputStream inputStream, java.lang.String systemId)
```

Construct a StreamSource from a byte stream. Normally, a stream should be used rather than a reader, so that the XML parser can resolve character encoding specified by the XML declaration.

This constructor allows the systemID to be set in addition to the input stream, which allows relative URIs to be processed.

Parameters:

`inputStream` - A valid InputStream reference to an XML stream.

`systemId` - Must be a String that conforms to the URI syntax.

StreamSource(Reader)

```
public StreamSource(java.io.Reader reader)
```

Construct a StreamSource from a character reader. Normally, a stream should be used rather than a reader, so that the XML parser can resolve character encoding specified by the XML declaration. However, in many cases the encoding of the input stream is already resolved, as in the case of reading XML from a StringReader.

Parameters:

`reader` - A valid Reader reference to an XML character stream.

StreamSource(Reader, String)

```
public StreamSource(java.io.Reader reader, java.lang.String systemId)
```

Construct a StreamSource from a character reader. Normally, a stream should be used rather than a reader, so that the XML parser may resolve character encoding specified by the XML declaration. However, in many cases the encoding of the input stream is already resolved, as in the case of reading XML from a StringReader.

Parameters:

`reader` - A valid Reader reference to an XML character stream.

`systemId` - Must be a String that conforms to the URI syntax.

StreamSource(String)

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

Construct a StreamSource from a URL.

Parameters:

`systemId` - Must be a String that conforms to the URI syntax.

Methods

---

getInputStream()

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

Get the byte stream that was set with setByteStream.

Returns: The byte stream that was set with setByteStream, or null if setByteStream or the ByteStream constructor was not called.

---

getPublicId()

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

Get the public identifier that was set with setPublicId.

Returns: The public identifier that was set with setPublicId, or null if setPublicId was not called.

---

getReader()

```
public java.io.Reader getReader()
```

Get the character stream that was set with setReader.

Returns: The character stream that was set with setReader, or null if setReader or the Reader constructor was not called.

---

getSystemId()

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

Get the system identifier that was set with setSystemId.

Specified By: `getSystemId()` in interface `Source`

Returns: The system identifier that was set with setSystemId, or null if setSystemId was not called.

---

setInputStream(InputStream)

```
public void setInputStream(java.io.InputStream inputStream)
```

Set the byte stream to be used as input. Normally, a stream should be used rather than a reader, so that the XML parser can resolve character encoding specified by the XML declaration.

If this Source object is used to process a stylesheet, normally setSystemId should also be called, so that relative URL references can be resolved.

Parameters:
`inputStream` - A valid InputStream reference to an XML stream.

---

setPublicId(String)

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

Set the public identifier for this 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.

setReader(Reader)

```
public void setReader(java.io.Reader reader)
```

Set the input to be a character reader. Normally, a stream should be used rather than a reader, so that the XML parser can resolve character encoding specified by the XML declaration. However, in many cases the encoding of the input stream is already resolved, as in the case of reading XML from a StringReader.

Parameters:

`reader` - A valid Reader reference to an XML CharacterStream.

setSystemId(File)

```
public void setSystemId(java.io.File f)
```

Set the system ID from a File reference.

Parameters:

`f` - Must a non-null File reference.

setSystemId(String)

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

Set the system identifier for this 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).

Specified By: `setSystemId(String)` in interface `Source`

Parameters:

`systemId` - The system identifier as a URL string.

SECTION 5

Conformance Requirements

This section describes the conformance requirements for parser implementations of this specification. Parser implementations that are accessed via the APIs defined here must implement these constraints, without exception, to provide a predictable environment for application development and deployment.

Note that applications may provide non-conformant implementations that are able to support the plugability mechanism defined in the specification, however the system default processor must meet the conformance requirements defined below.

All implementations of this specification need to be conformant as per Section 5 of the XML 1.0 recommendation (second edition) (http://www.w3.org/TR/2000/REC-xml-20001006#sec-conformance), Section 6 of the XML Namespaces recommendation (http://www.w3.org/TR/REC-xml-names/) and Section 17 of the XSLT recommendation (http://www.w3.org/TR/xslt). Parsers that support validation, only need to support DTDs. In addition to the above, implementations of the SAX 2.0, SAX2.0 extensions and DOM Level 2 core interfaces must be supported.

SECTION 6

Change History

This section lists the changes that have occurred over the development of this specification.

6.1 From 1.1 Public Review 1 to 1.1 Public Review 2

Modified javax.xml.transform to include a new set of APIs for transformation.

Changed endorsed specifications section to include the second edition of the XML 1.0 recommendation, the DOM Level2 core recommendation and the final version of SAX2-extensions.

6.2 From 1.0 Final Release to 1.1 Public Review

Added parameter systemId to all the parse and transform methods which take Streams as parameters. This was done to provide a base to resolve relative URIs.

Added setIgnoreWhitespace, setExpandEntityReference, setIgnoringComments, setAttributes and the corresponding getters to DocumentBuilderFactory.

Added get/setAttribute to TransformFactory.

Added setEntityResolver, setErrorHandler and get/setXSLParam to Transform.

Added get/setFeature to SAXParserFactory.

Added get/setProperty to SAXParser.

Added SAX2 extensions.

Added Transformations

Added more mechanisms to look up an implementation of the various factories..

Removed conformance requirements from the specification and just refer to the conformance requirements as required by the specifications included by reference.

6.3 From 1.0 Public Release to 1.0 Final Release

The reservation of the java and javax namespace prefixes was removed. The XML Namespace specification is clear that a namespace is a collection of names that is identified by a URI reference. The prefix is a local identifier for the URI reference, therefore the reservation of the java and javax namespaces was in error.

6.4 From 1.0 Public Review to 1.0 Public Release

From the Public Review draft of this specification to the Public Release version, the specification was reordered and rewritten to address general feedback from the user community. This feedback indicated that the specification was too detailed in describing the endorsed specifications and not detailed enough in describing the plugability layer.

The `newParser` method of the `SAXParserFactory` abstract class was removed. Feedback showed that it was confusing to be able to obtain both the `SAXParser` wrapper and the underlying implementation from the factory. Removing this method allows the API to be more understandable while preserving the ability to access the underlying parser via the `getParser` method of the `SAXParser` abstract class.

The `getLocale` and `setLocale` methods of the various classes were removed. Instead it was felt that parser implementation authors should report errors in the configured default locale of the execution environment.

A new exception named `ParserConfigurationException` was added so that a parser factory can signal to an application that it can’t provide a parser with the desired configuration. The `checkXXX` methods aren’t sufficient for this pur-

pose as a situation may arise where there is a mutually exclusive setting of various parser properties. At this time, this problem is potentially minor as there are only two settable properties on each of the parser types, but in the future as the number of settable properties increases, the problem would get harder to solve without an exception that could be thrown at parser creation time. As part of this change, the `setXXX` property methods of the factories no longer throw an `IllegalArgumentException` if they are set to a property which cannot be supported.

The `FactoryException` class was renamed to `FactoryConfigurationError`. This rename was undertaken to emphasize that such an error condition is a fatal condition that an application should not be reasonable expected to handle.

SECTION 7

Future Directions

This version of the Java API for XML Processing includes the basic facilities for working with XML documents using either the SAX, DOM and XSLT APIs. However, there is always more to be done.

This section briefly describes our plans for future versions of this specification. Please keep in mind that the items listed here are preliminary and there is no commitment to the inclusion of any specific feature in any specific version of the specification. In addition, this list of items is by no means the only features that may appear in a future revision. Your feedback is encouraged.

7.1 Updated SAX and DOM Support

As future versions of SAX and DOM evolve it will be incorporated into the future version of this API.

7.2 Update XSL Plugability Support

XSL (eXtensible Stylesheet Language) is a language for expressing stylesheets that can be used with XML document. It consists of two parts:

- A language for transforming XSL documents (also known as XSLT)
- An XML vocabulary for specifying formatting specfics

XSL Transformations has been formalized as a W3C Recommendation. In a future version of the specification, we would like to provide a plugability API to allow an application programmer to provide an XML document and an XSLT document to a wrapped XSLT processor and obtain a transformed result.

7.3 Plugability Mechanism Enhancements

Various ways of making the plugability mechanism work have been incorporated into this version of the spec. However if there are other ways in the future to enhance this, it will be included in the future versions of this API.