22 - Nov - 08

ALL are **GETTER** methods ONLY

30 - Dec - 08

jav events

Interface Attribute

3 methods

All Superinterfaces:

XMLEvent, XMLStreamConstants

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

All Known Subinterfaces:

Namespace

Secondary event cannot read directly from XMLEventreader. it has to read from StartElement. getAttributeByName() / getAttributes()

```
public interface Attribute
```

extends XMLEvent

An interface that contains information about an attribute. Attributes are reported as a set of events accessible from a StartElement. Other applications may report Attributes as first-order events, for example as the results of an XPath expression.

Huugum?!!?

Since:

1.6

Version:

1.0

Author:

See Also:

`StartElement`

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants

ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

java.lang.String	**~~getDTDType~~**() ~~Gets the type of~~ this attribute, default is the String "CDATA"
QName	**getName**() ~~Returns the QName for this attribute~~
java.lang.String	**getValue**() ~~Gets the normalized value of this attribut~~
boolean	**isSpecified**() ~~A flag indicating whether this attribute was actually specified in the start-tag of its element, or was defaulted from the schema.~~

this will be FALSE if the Attribute Event created on memory not from file reading

Methods inherited from interface javax.xml.stream.events.XMLEvent

asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getName

QName **getName**()

~~Returns the QName for this attribute~~

---

getValue

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

Gets the normalized value of this attribute

---

getDTDType

~~`java.lang.String` **`getDTDType`**`()`~~

~~Gets the type of this attribute, default is the String "CDATA"~~

~~**Returns:**~~
~~the type as a String, default is "CDATA"~~

---

isSpecified

`boolean` **`isSpecified`**`()`

~~A flag indicating whether this attribute was actually specified in the start-tag of its element, or was defaulted from the schema.~~

~~**Returns:**~~
~~returns true if this was specified in the start element~~

---

22 - Nov - 08

30 - Dec - 08

nothing

java vents

Interface EndDocument

All Superinterfaces:

XMLEvent, XMLStreamConstants

```
public interface EndDocument
```

extends XMLEvent

A marker interface for the end of the document

Since:

1.6

Version:

1.0

Author:

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

22 - Nov - 08

30 - Dec - 08

j n.events

Interface EndElement

2 methods

All Superinterfaces:

XMLEvent, XMLStreamConstants

ALL are **GETTER** methods ONLY

```
public interface EndElement
```

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

extends XMLEvent

~~An interface for the end element event. An EndElement is reported for each End Tag in the document.~~

Since:

1.6

Version:

1.0

Author:

See Also:

XMLEvent

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

QName	**getName**() ~~Get the name of this event~~
java.util.Iterator	**getNamespaces**() ~~Returns an Iterator of namespaces that have gone out of scope~~.

Methods inherited from interface javax.xml.stream.events.XMLEvent

asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getName

QName **getName**()

~~Get the name of this event~~

~~**Returns:**~~

~~the qualified name of this event~~

---

getNamespaces

it return iter object. but it dont have any namespace. even it has some namespaces

java.util.Iterator **getNamespaces**()

Returns an Iterator of namespaces that have gone out of scope. Returns an empty iterator if no namespaces have gone out of scope.

```
Iterator iter = endElement.getNamespaces();
  while(iter.hasNext()) {
   logger.info("<>"+iter.next());
  }
```

Returns:

an Iterator over Namespace interfaces, or an empty iterator

21 - Nov - 08

31 - Dec - 08

1 methods

javax.xml.stream

Interface EventFilter

Before to study this class first finish, XMLEventFactory and its individual Events

```
public interface EventFilter
```

This interface declares a simple filter interface that one can create to filter XMLEventReaders

Since:
1.6

Version:
1.0

Author:

Method Summary

boolean	**accept**(XMLEvent event) Tests whether this event is part of this stream.

Method Detail

accept

```
boolean accept(XMLEvent event)
```

~~Tests whether this event is part of this stream. This method will return true if this filter accepts this event and false otherwise.~~

Parameters:

`event` - the event to test

Returns:

true if this filter accepts this event, false otherwise

---

22 - Nov - 08

ALL are **GETTER** methods ONLY

30 - Dec - 08

ja .events

Interface Characters

3 methods

3 are boolean methods

All Superinterfaces:

XMLEvent, XMLStreamConstants

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

`public interface` **`Characters`**

extends XMLEvent

This describes the interface to Characters events. All text events get reported as Characters events. Content, CData and whitespace are all reported as Characters events. ~~IgnorableWhitespace, in most cases, will be set to false unless an element declaration of element content is present for the current element.~~

Ignorable Whitespace is DTD related

Since:

1.6

Version:

1.0

Author:

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

java.lang.String	**getData**() ~~Get the character data of this event~~
boolean	**isCData**() ~~Returns true if this is a CData section.~~
boolean	**isIgnorableWhiteSpace**() ~~Return true if this is ignorableWhiteSpace.~~
boolean	**isWhiteSpace**() ~~Returns true if this set of Characters is all whitespace.~~ Super

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getData

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

~~Get the character data of this event~~

---

isWhiteSpace

`boolean` **`isWhiteSpace`**`()`

Returns true if this set of Characters is all whitespace. Whitespace inside a document is reported as CHARACTERS. This method allows checking of CHARACTERS events to see if they are

javax.xml.stream.util

Class EventReaderDelegate

```
java.lang.Object
  └ javax.xml.stream.util.EventReaderDelegate
```

All Implemented Interfaces:

java.util.Iterator, XMLEventReader

```
public class EventReaderDelegate
```

extends java.lang.Object
implements XMLEventReader

This is the base class for deriving an XMLEventReader filter. This class is designed to sit between an XMLEventReader and an application's XMLEventReader. By default each method does nothing but call the corresponding method on the parent interface.

Since:

1.6

Version:

1.0

Author:

See Also:

`XMLEventReader`, `StreamReaderDelegate`

Constructor Summary

EventReaderDelegate() Construct an empty filter with no parent.
EventReaderDelegate(XMLEventReader reader) Construct an filter with the specified parent.

Method Summary

void	**close**() Frees any resources associated with this Reader.
java.lang.String	**getElementText**() Reads the content of a text-only element.
XMLEventReader	**getParent**() Get the parent of this instance.
java.lang.Object	**getProperty**(java.lang.String name) Get the value of a feature/property from the underlying implementation
boolean	**hasNext**() Check if there are more events.
java.lang.Object	**next**()
XMLEvent	**nextEvent**() Get the next XMLEvent
XMLEvent	**nextTag**() Skips any insignificant space events until a START_ELEMENT or END_ELEMENT is reached.
XMLEvent	**peek**() Check the next XMLEvent without reading it from the stream.
void	**remove**()
void	**setParent**(XMLEventReader reader) Set the parent of this instance.

Methods inherited from class java.lang.Object

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

Constructor Detail

EventReaderDelegate

`public` **`EventReaderDelegate`**`()`

Construct an empty filter with no parent.

EventReaderDelegate

`public` **`EventReaderDelegate`**`(XMLEventReader reader)`

Construct an filter with the specified parent.

Parameters:
`reader` - the parent

Method Detail

setParent

`public void` **`setParent`**`(XMLEventReader reader)`

Set the parent of this instance.

Parameters:
`reader` - the new parent

---

getParent

composed of only whitespace characters

---

isCData

`boolean` **`isCData()`**

Returns true if this is a CData section. If this event is CData its event type will be CDATA If javax.xml.stream.isCoalescing is set to true CDATA Sections that are surrounded by non CDATA characters will be reported as a single Characters event. This method will return false in this case.

~~isIgnorableWhiteSpace~~

~~`boolean` **`isIgnorableWhiteSpace()`**~~

~~Return true if this is ignorableWhiteSpace. If this event is ignorableWhiteSpace its event type will be SPACE.~~

`public XMLEventReader getParent()`

Get the parent of this instance.

Returns:
the parent or null if none is set

---

nextEvent

`public XMLEvent nextEvent() throws XMLStreamException`

Description copied from interface: XMLEventReader
Get the next XMLEvent

Specified by:
`nextEvent` in interface `XMLEventReader`

Throws:
`XMLStreamException` - if there is an error with the underlying XML.

See Also:
`XMLEvent`

---

next

`public java.lang.Object next()`

Specified by:
`next` in interface `java.util.Iterator`

---

hasNext

`public boolean hasNext()`

Description copied from interface: `XMLEventReader`

Check if there are more events. Returns true if there are more events and false otherwise.

Specified by:
`hasNext` in interface `java.util.Iterator`

Specified by:
`hasNext` in interface `XMLEventReader`

Returns:
true if the event reader has more events, false otherwise

---

peek

```
public XMLEvent peek()
              throws XMLStreamException
```

Description copied from interface: `XMLEventReader`

Check the next XMLEvent without reading it from the stream. Returns null if the stream is at EOF or has no more XMLEvents. A call to peek() will be equal to the next return of next().

Specified by:
`peek` in interface `XMLEventReader`

Throws:
`XMLStreamException`

See Also:
`XMLEvent`

---

close

```
public void close()
           throws XMLStreamException
```

Description copied from interface: `XMLEventReader`

Frees any resources associated with this Reader. This method does not close the underlying input source.

Specified by:
`close` in interface `XMLEventReader`

Throws:
`XMLStreamException` - if there are errors freeing associated resources

---

getElementText

```
public java.lang.String getElementText()
                                throws XMLStreamException
```

Description copied from interface: `XMLEventReader`

Reads the content of a text-only element. Precondition: the current event is START_ELEMENT. Postcondition: The current event is the corresponding END_ELEMENT.

Specified by:
`getElementText` in interface `XMLEventReader`

Throws:
`XMLStreamException` - if the current event is not a START_ELEMENT or if a non text element is encountered

---

nextTag

```
public XMLEvent nextTag()
                 throws XMLStreamException
```

Description copied from interface: `XMLEventReader`

Skips any insignificant space events until a START_ELEMENT or END_ELEMENT is reached. If anything other than space characters are encountered, an exception is thrown. This method should be used when processing element-only content because the parser is not able to recognize ignorable whitespace if the DTD is missing or not interpreted.

Specified by:
`nextTag` in interface `XMLEventReader`

Throws:
`XMLStreamException` - if anything other than space characters are encountered

---

getProperty

```
public java.lang.Object getProperty(java.lang.String name)
                                throws java.lang.IllegalArgumentException
```

Description copied from interface: `XMLEventReader`

Get the value of a feature/property from the underlying implementation

Specified by:

`getProperty` in interface `XMLEventReader`

Parameters:

`name` - The name of the property

Returns:

The value of the property

Throws:

`java.lang.IllegalArgumentException` - if the property is not supported

---

remove

```
public void remove()
```

Specified by:

`remove` in interface `java.util.Iterator`

---

22 - Nov - 08

ALL are **GETTER** methods ONLY

30 - Dec - 08

ja vents

Interface Comment

1 method

All Superinterfaces:

XMLEvent, XMLStreamConstants

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

```
public interface Comment
```

extends XMLEvent

it extends XMLEvent but **NOT** Characters as DOM API

An interface for comment events

Since:

1.6

Version:

1.0

Author:

toString() method of this class append <! , !> symbol but, getText() method return only text

the same behavior for CDATA also

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

java.lang.String	**getText**() this method, will return ONLY text of this comment WITHOUT <!-- --> Return the string data of the comment, returns empty string if it does not exist

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getText

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

Return the string data of the comment, returns empty string if it does not exist

15 - Nov - 08

27 - Dec - 08

5 methods

jav

Interface Location

we cannot use this object as SAX's Locator getting once and access any time.
here we have to get this location object at EVERY time to know the exact location

```
public interface Location
```

Provides information on the location of an event. All the information provided by a Location is optional. For example an application may only report line numbers.

Since:
1.6

Version:
1.0

Author:

Method Summary

getCharacterOffset() and getColumnNumber() method will return same value ... ????

int	**getCharacterOffset**() Return the byte or character offset into the input source this location is pointing to.
int	**getColumnNumber**() Return the column number where the current event ends, returns -1 if none is available.
int	**getLineNumber**() Return the line number where the current event ends, returns -1 if none is available.
java.lang.String	**getPublicId**() Returns the public ID of the XML
java.lang.String	**getSystemId**() Returns the system ID of the XML

Method Detail

getLineNumber

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

~~Return the line number where the current event ends, returns -1 if none is available~~.

Returns:
~~the current line number~~

---

getColumnNumber

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

~~Return the column number where the current event ends, returns -1 if none is available~~.

Returns:
the current column number

---

getCharacterOffset

`int` **`getCharacterOffset`**`()`

Return the byte or character offset into the input source this location is pointing to. If the input source is a file or a byte stream then this is the byte offset into that stream, but if the input source is a character media then the offset is the character offset. Returns -1 if there is no offset available.

Returns:
the current offset

---

getPublicId

public id means, web url

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

Returns the public ID of the XML

Returns:
the public ID, or null if not available

---

getSystemId

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

Returns the system ID of the XML

Returns:
the system ID, or null if not available

---

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

22 - Nov - 08

30 - Dec - 08

j m.events

Interface Namespace

The StAX parser maintains a namespace stack, which holds information about all XML namespaces defined for the current element and its ancestors. The namespace stack is exposed through the **javax.xml.namespace. NamespaceContext** interface, and can be accessed by namespace prefix or URI

All Superinterfaces:

Attribute, XMLEvent, XMLStreamConstants

ALL are **GETTER** methods ONLY

```
public interface Namespace
```

3 methods

extends Attribute

An interface that contains information about a namespace. Namespaces are accessed from a StartElement.

it is cannot read from XMLEventReader. it has to get from StartElement. getNamespace() / getNamesapceUri()

Since:

1.6

Version:

1.0

Author:

See Also:

`StartElement`

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

java.lang.String	**getNamespaceURI**() Gets the uri bound to the prefix of this namespace
java.lang.String	**getPrefix**() Gets the prefix, returns "" if this is a default namespace declaration.
boolean	**isDefaultNamespaceDeclaration**() returns true if this attribute declares the default namespace

Methods inherited from interface javax.xml.stream.events.Attribute
getDTDType, getName, getValue, isSpecified

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getPrefix

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

Gets the prefix, returns "" if this is a default namespace declaration.

---

getNamespaceURI

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

Gets the uri bound to the prefix of this namespace

---

isDefaultNamespaceDeclaration

`boolean` **`isDefaultNamespaceDeclaration()`**

if prefix is EMPTY, then this method return TRUE. i tested

returns true if this attribute declares the default namespace

---

---

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

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

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

Java API for XML Processing (JAXP) 1.4

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

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

since JAXP 1.3

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

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

it is Introduced by SUN

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

1 2 3 4 5

11 3 5 2 2

Package javax.xml.stream

StAX used Iterator
Design pattern

Interface Summary

ALL of StAX API has Great support of Namespace.

EventFilter	~~This inter ... to filter XMLEve~~
Location	~~Provides~~
StreamFilter	~~This inter ... to filter XMLStre~~
XMLEventReader	~~This is th~~
XMLEventWriter	~~This is th~~
XMLReporter	~~This inter~~
XMLResolver	~~This inter~~
XMLStreamConstants	~~This inter~~
XMLStreamReader	~~The XMLStreamReader interface allows forward, read-only access to XML.~~
XMLStreamWriter	~~The XMLStreamWriter interface specifies how to write XML.~~

Class Summary

XMLEventFactory	~~This interface defines a utility class for creating instances of XMLEvents~~
XMLInputFactory	~~Defines an abstract implementation of a factory for getting streams.~~
XMLOutputFactory	~~Defines an abstract implementation of a factory for getting XMLEventWriters and XMLStreamWriters.~~

Exception Summary

XMLStreamException	~~The base exception for unexpected processing errors.~~

Error Summary

FactoryConfigurationError	An error class for reporting factory configuration errors.

Comparing Cursor and Iterator APIs

Before choosing between the cursor and iterator APIs, you should note a few things that you can do with the iterator API that you cannot do with cursor API:

Objects created from the XMLEvent subclasses are immutable, and can be used in arrays, lists, and maps, and can be passed through your applications even after the parser has moved on to subsequent events.
You can create subtypes of XMLEvent that are either completely new information items or extensions of existing items but with additional methods.
You can add and remove events from an XML event stream in much simpler ways than with the cursor API.
Similarly, keep some general recommendations in mind when making your choice:

If you are programming for a particularly memory-constrained environment, like J2ME, you can make smaller, more efficient code with the cursor API.
If performance is your highest priority--for example, when creating low-level libraries or infrastructure--the cursor API is more efficient.
If you want to create XML processing pipelines, use the iterator API.
If you want to modify the event stream, use the iterator API.
If you want to your application to be able to handle pluggable processing of the event stream, use the iterator API.
In general, if you do not have a strong preference one way or the other, using the iterator API is recommended because it is more flexible and extensible, thereby "future-proofing" your applications.

22 - Nov - 08

Package javax.xml.stream.events

9 interfaces

Interface Summary	
Attribute	An interface that contains information about an attribute.
Characters	This describes the interface to Characters events.
Comment	An interface for comment events
~~**DTD**~~	~~This is the top level interface for events dealing with DTDs~~
EndDocument	A marker interface for the end of the document
EndElement	An interface for the end element event.
~~**EntityDeclaration**~~	~~An interface for handling Entity Declarations This interface is used to record and report unparsed entity declarations.~~
~~**EntityReference**~~	~~An interface for handling Entity events.~~
Namespace	An interface that contains information about a namespace.
~~**NotationDeclaration**~~	~~An interface for handling Notation Declarations Receive notification of a notation declaration event.~~
~~**ProcessingInstruction**~~	~~An interface that describes the data found in processing instructions~~
StartDocument	An interface for the start document event
StartElement	The StartElement interface provides access to information about start elements.
XMLEvent	This is the base event interface for handling markup events.

No need to work out. just read.

none of these classes / interface has direct usage with application,
i hope i dont want , i just skimmed these

Package javax.xml.stream.util

Interface Summary

XMLEventAllocator	This interface defines a class that allows a user to register a way to allocate events given an XMLStreamReader.
XMLEventConsumer	This interface defines an event consumer interface.

for FILTER related to stream and event reader
NO NEED TO LEARN

Class Summary

EventReaderDelegate	This is the base class for deriving an XMLEventReader filter.
StreamReaderDelegate	This is the base class for deriving an XMLStreamReader filter This class is designed to sit between an XMLStreamReader and an application's XMLStreamReader.

that is one stream reader and filtered one more stream reader

22 - Nov - 08

ja 30 - Dec - 08 **.events**

Interface StartDocument

4 methods

ALL are **GETTER** methods ONLY

All Superinterfaces:

XMLEvent, XMLStreamConstants

ALL are **GETTER** methods ONLY

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

```
public interface StartDocument
```

extends XMLEvent

~~An interface for the start document event~~

Since:

1.6

Version:

1.0

Author:

Field ~~Summary~~

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

so, this method will be true, only if it is read from XML document

boolean	**encodingSet()** Returns true if CharacterEncodingScheme was set in the encoding declaration of the document
java.lang.String	**getCharacterEncodingScheme()** from factory Returns the encoding style of the XML data UTF-8|16
java.lang.String	**getSystemId()** Returns the system ID of the XML data
java.lang.String	**getVersion()** from factory Returns the version of XML of this XML stream
~~boolean~~	~~**isStandalone()**~~ ~~Returns if this XML is standalone~~ for DTD only
~~boolean~~	~~**standaloneSet()**~~ ~~Returns true if the standalone attribute was set in the encoding declaration of the document.~~

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getSystemId

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

~~Returns the system ID of the XML data~~

Returns:
the system ID, defaults to "" but not NULL

---

getCharacterEncodingScheme

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

~~Returns the encoding style of the XML data~~

Returns:
the character encoding, defaults to "UTF-8"

---

encodingSet

they would have named like isEncodingSpecifed() instead of StartElement.encodingSet()

`boolean` **`encodingSet()`**

Returns true if CharacterEncodingScheme was set in the encoding declaration of the document

---

~~isStandalone~~

~~`boolean` **`isStandalone()`**~~

~~Returns if this XML is standalone~~

~~**Returns:**~~
~~the standalone state of XML, defaults to "no"~~

~~standaloneSet~~

~~`boolean` **`standaloneSet()`**~~

~~Returns true if the standalone attribute was set in the encoding declaration of the document~~.

---

getVersion

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

Returns the version of XML of this XML stream

Returns:
the version of XML, defaults to "1.0"

Overview **Package** **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES** **All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

22 - Nov - 08

30 - Dec - 08

6 methods
3 - namespace related
2 - attribute related
1 - getter for name of startelement

ALL are **GETTER** methods ONLY

jav events

Interface StartElement

All Superinterfaces:

XMLEvent, XMLStreamConstants

```
public interface StartElement
```

extends XMLEvent

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

~~The StartElement interface provides access to information about start elements. A StartElement is reported for each Start Tag in the document.~~

Since:

1.6

Version:

1.0

Author:

since this is STARTelement and not just ELEMENT, StartElement does not associated with its text.
element text will be separate xml event. so, dont expect method to retrieve text of element here.

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

Attribute	**getAttributeByName**(QName name) Returns the attribute referred to by this name
java.util.Iterator	**getAttributes**() Returns an Iterator of non-namespace declared attributes declared on this START_ELEMENT, returns an empty iterator if there are no attributes.
QName	**getName**() Get the name of this event
NamespaceContext	**getNamespaceContext**() Gets a read-only namespace context.
java.util.Iterator	**getNamespaces**() Returns an Iterator of namespaces declared on this element.
java.lang.String	**getNamespaceURI**(java.lang.String prefix) Gets the value that the prefix is bound to in the context of this element.

every start element , can have its own NamespaceContext

Methods inherited from interface javax.xml.stream.events.XMLEvent
asCharacters, asEndElement, asStartElement, getEventType, getLocation, getSchemaType, isAttribute, isCharacters, isEndDocument, isEndElement, isEntityReference, isNamespace, isProcessingInstruction, isStartDocument, isStartElement, writeAsEncodedUnicode

Method Detail

getName

QName **getName**()

Get the name of this event

Returns:
the qualified name of this event

except getName() method, all others are MAY return NULL or EMPTY,

ALL of these are parameter while creating instance for StartElement

---

getAttributes

`java.util.Iterator` **`getAttributes`**`()`

Returns an Iterator of non-namespace declared attributes declared on this START_ELEMENT, returns an empty iterator if there are no attributes. The iterator must contain only implementations of the javax.xml.stream.Attribute interface. Attributes are fundamentally unordered and may not be reported in any order.

Returns:

a readonly Iterator over Attribute interfaces, or an empty iterator

---

getNamespaces

`java.util.Iterator` **`getNamespaces`**`()`

Returns an Iterator of namespaces declared on this element. This Iterator does not contain previously declared namespaces unless they appear on the current START_ELEMENT. Therefore this list may contain redeclared namespaces and duplicate namespace declarations. Use the getNamespaceContext() method to get the current context of namespace declarations.

The iterator must contain only implementations of the javax.xml.stream.Namespace interface.

A Namespace isA Attribute. One can iterate over a list of namespaces as a list of attributes. However this method returns only the list of namespaces declared on this START_ELEMENT and does not include the attributes declared on this START_ELEMENT. Returns an empty iterator if there are no namespaces.

good

Returns:

a readonly Iterator over Namespace interfaces, or an empty iterator

---

getAttributeByName

`Attribute` **`getAttributeByName`**`(QName name)`

~~Returns the attribute referred to by this name~~

~~**Parameters:**~~

`name` ~~- the qname of the desired name~~

~~**Returns:**~~

~~the attribute corresponding to the name value or null~~

---

getNamespaceContext

`NamespaceContext` **`getNamespaceContext()`**

Gets a read-only namespace context. If no context is available this method will return an empty namespace context. The NamespaceContext contains information about all namespaces in scope for this StartElement.

so, NamespaceContext is just collection of NameSpace object for a particular START_ELEMENT

Returns:

the current namespace context

---

getNamespaceURI

`java.lang.String` **`getNamespaceURI`**`(java.lang.String prefix)`

Gets the value that the prefix is bound to in the context of this element. Returns null if the prefix is not bound in this context

Parameters:

`prefix` - the prefix to lookup

Returns:

the uri bound to the prefix or null

---

22 - Nov - 08

javax 30 - Dec - 08 **nts**

Interface XMLEvent

14 methods

7 - HANDY methods to avoid exception scanario (boolean)
3 - as event methods

All Superinterfaces:

XMLStreamConstants

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

All Known Subinterfaces:

Attribute, Characters, Comment, ~~DTD~~, EndDocument, EndElement, ~~EntityDeclaration~~, ~~EntityReference~~, Namespace, ~~NotationDeclaration~~, ~~ProcessingInstruction~~, StartDocument, StartElement

These UTILITY method will be useful for XMLEventReader to check what is the current event

---

```
public interface XMLEvent
```

extends XMLStreamConstants

toString() method of each XMLEvent object overridden appropriately.
it is good design, directly write into underlaying stream without any special code

This is the base event interface for handling markup events. Events are value objects that are used to communicate the XML 1.0 InfoSet to the Application. Events may be cached and referenced after the parse has completed.

Yes great feature

Since:

1.6

Version:

1.0

Author:

Copyright (c) 20 served.

See Also:

XMLEventRea gInstruction, StartElement, EndElement, t, EntityReference, EntityDecla on

EVENT VALUE

START_ELEMENT = 1
END_ELEMNET = 2

CHARACTERS = 4 (CDATA / text)

COMMENT = 5

START_DOCUMENT = 7
END_DOCUMENT = 8

ATTRIBUTE = 10

NAMESPACE = 13

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants
ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

Characters	**asCharacters**() so, check isCharacters() method of this class before to call Returns this event as Characters, may result in a class cast exception if this event is not Characters.
EndElement	**asEndElement**() so, check isEndElement() method of this class before to call Returns this event as an end element event, may result in a class cast exception if this event is not a end element.
StartElement	**asStartElement**() so, check isStartElement() method of this class before to call Returns this event as a start element event, may result in a class cast exception if this event is not a start element.
int	**getEventType**() Returns an integer code for this event. this will be null unless we set in XMLEventFactory
Location	**getLocation**() Return the location of this event. location object will be usefull with XMLEventReader
QName	**getSchemaType**() This method is provided for implementations to provide optional type information about the associated event. where can i associate an xml schema ? can i do via xml file itsef
boolean	**isAttribute**() A utility function to check if this event is an Attribute.
boolean	**isCharacters**() 2 A utility function to check if this event is Characters.
boolean	**isEndDocument**() 3 A utility function to check if this event is an EndDocument.

boolean	**isEndElement**() A utility function to check if this event is a EndElement.
boolean	**isEntityReference**() A utility function to check if this event is an EntityReference.
boolean	**isNamespace**() A utility function to check if this event is a Namespace.
boolean	**isProcessingInstruction**() A utility function to check if this event is a ProcessingInstruction.
boolean	**isStartDocument**() A utility function to check if this event is a StartDocument.
boolean	**isStartElement**() A utility function to check if this event is a StartElement.
void	**writeAsEncodedUnicode**(java.io.Writer writer) This method will write the XMLEvent as per the XML 1.0 specification as Unicode characters.

this is great feature to force us to use Iterator API :)

super method to write xml content into unicode ... super.

Method Detail

but, this does not work. it written nothing in output stream

getEventType

`int` **getEventType**()

Returns an integer code for this event.

See Also:

XMLStreamConstants.START_ELEMENT, XMLStreamConstants.END_ELEMENT, XMLStreamConstants.CHARACTERS, XMLStreamConstants.ATTRIBUTE, XMLStreamConstants.NAMESPACE, XMLStreamConstants.PROCESSING_INSTRUCTION, XMLStreamConstants.COMMENT, XMLStreamConstants.START_DOCUMENT, XMLStreamConstants.END_DOCUMENT, XMLStreamConstants.DTD

getLocation

`Location` **`getLocation()`**

~~Return the location of this event. The Location returned from this method is non-volatile and will retain its information.~~

See Also:
`Location`

---

isStartElement

`boolean` **`isStartElement()`**

~~A utility function to check if this event is a StartElement~~.

See Also:
`StartElement`

---

isAttribute

2

`boolean` **`isAttribute()`**

A ~~utility function to check if this event is an Attribute.~~

See Also:
`Attribute`

---

isNamespace

3

`boolean` **`isNamespace()`**

A ~~utility function to check if this event is a Namespace.~~

See Also:
Namespace

---

isEndElement

`boolean` **`isEndElement()`**

4

~~A utility function to check if this event is a EndElement~~.

See Also:
EndElement

~~isEntityReference~~

~~`boolean` **`isEntityReference()`**~~

~~A utility function to check if this event is an EntityReference.~~

~~**See Also:**~~
EntityReference

~~isProcessingInstruction~~

~~`boolean` **`isProcessingInstruction()`**~~

~~A utility function to check if this event is a ProcessingInstruction.~~

~~**See Also:**~~
ProcessingInstruction

---

isCharacters

5

`boolean` **`isCharacters()`**

~~A utility function to check if this event is Characters~~.

See Also:
`Characters`

---

isStartDocument

`boolean` **`isStartDocument()`**

6

A ~~utility function to check if this event is a StartDocument~~.

See Also:
`StartDocument`

---

isEndDocument

`boolean` **`isEndDocument()`**

7

~~A utility function to check if this event is an EndDocument~~.

See Also:
`EndDocument`

---

asStartElement

`StartElement` **`asStartElement()`**

A

check with isStartElement() method of this class before to use this mtd

Returns this event as a start element event, may result in a class cast exception if this event is not a start element.

---

asEndElement

B

check with isEndElement() method of this class before to use this mtd

`EndElement` **`asEndElement()`**

Returns this event as an end element event, may result in a class cast exception if this event is not a end element.

check with isCharacters () method of this class before to use this mtd

---

asCharacters

so, check isCharacters() method of this class before to call

`Characters` **`asCharacters()`**

C

Returns this event as Characters, may result in a class cast exception if this event is not Characters.

---

getSchemaType

`QName` **`getSchemaType()`**

This method is provided for implementations to provide optional type information about the associated event. It is optional and will return null if no information is available.

---

writeAsEncodedUnicode

```
void writeAsEncodedUnicode(java.io.Writer writer)
                           throws XMLStreamException
```

This method will write the XMLEvent as per the XML 1.0 specification as Unicode characters. No indentation or whitespace should be outputted. Any user defined event type SHALL have this method called when being written to on an output stream. Built in Event types MUST implement this method, but implementations MAY choose not call these methods for optimizations reasons when writing out built in Events to an output stream. The output generated MUST be equivalent in terms of the infoset expressed.

but, it was not worked for me. i tried...at 4th round also

Parameters:

`writer` - The writer that will output the data

Throws:

`XMLStreamException` - if there is a fatal error writing the event

---

javax.xml.stream.util

Interface XMLEventAllocator

`public interface` **`XMLEventAllocator`**

This interface defines a class that allows a user to register a way to allocate events given an XMLStreamReader. An implementation is not required to use the XMLEventFactory implementation but this is recommended. The XMLEventAllocator can be set on an XMLInputFactory using the property "javax.xml.stream.allocator"

Since:
1.6
Version:
1.0
Author:

See Also:
`XMLInputFactory`, `XMLEventFactory`

Method Summary

`XMLEvent`	**`allocate`**`(XMLStreamReader reader)` This method allocates an event given the current state of the XMLStreamReader.
`void`	**`allocate`**`(XMLStreamReader reader, XMLEventConsumer consumer)` This method allocates an event or set of events given the current state of the XMLStreamReader and adds the event or set of events to the consumer that was passed in.

XMLEventAllocator	**newInstance**() This method creates an instance of the XMLEventAllocator.

Method Detail

newInstance

XMLEventAllocator **newInstance**()

This method creates an instance of the XMLEventAllocator. This allows the XMLInputFactory to allocate a new instance per reader.

---

allocate

XMLEvent **allocate**(XMLStreamReader reader)
throws XMLStreamException

This method allocates an event given the current state of the XMLStreamReader. If this XMLEventAllocator does not have a one-to-one mapping between reader states and events this method will return null. This method must not modify the state of the XMLStreamReader.

Parameters:
`reader` - The XMLStreamReader to allocate from

Returns:
the event corresponding to the current reader state

Throws:
XMLStreamException

---

allocate

void **allocate**(XMLStreamReader reader,
XMLEventConsumer consumer)
throws XMLStreamException

This method allocates an event or set of events given the current state of the XMLStreamReader and adds the event or set of events to the consumer that was passed in. This method can be used to expand or contract reader states into event states. This method may modify the state of the XMLStreamReader.

Parameters:

`reader` - The XMLStreamReader to allocate from

`consumer` - The XMLEventConsumer to add to.

Throws:

`XMLStreamException`

---

javax.xml.stream.util

Interface XMLEventConsumer

All Known Subinterfaces:

XMLEventWriter

`public interface` **`XMLEventConsumer`**

This interface defines an event consumer interface. The contract of the of a consumer is to accept the event. This interface can be used to mark an object as able to receive events. Add may be called several times in immediate succession so a consumer must be able to cache events it hasn't processed yet.

Since:

1.6

Version:

1.0

Author:

Method Summary

`void`	**`add`**`(XMLEvent event)` This method adds an event to the consumer.

Method Detail

add

```
void add(XMLEvent event)
         throws XMLStreamException
```

This method adds an event to the consumer. Calling this method invalidates the event parameter. The client application should discard all references to this event upon calling add. The behavior of an application that continues to use such references is undefined.

Parameters:

`event` - the event to add, may not be null

Throws:

`XMLStreamException`

---

28 - Dec - 08

in XMLEventFactory no any configuration or set property / feature as other factories

javax.xml.stream

Class XMLEventFactory

24 methods

3 - ATTRIBUTE related methods
4 - CHARACTER related methods
7 - ELEMENT related methods
5 - DOCUMENT related methods
2 - NAMESPACE related methods
3 - WRITER related methods

```
java.lang.
  └javax.                    ry
```

All of XMLEvent object has over ride its toString() method to add special symbol like <, [![CDATA, <!--

```
public abstract class XM
```

usually domain object will have both setter / getter methods. but all of XML event object has Getter methods only.
So, setting value only done via factory methods

extends java.lang.Object

This interface defines a utility class for creating instances of XMLEvents

Since:
1.6
Version:
1.0
Author:

ALL of its methods are just create different Event object with overloaded.
no methods for any other purpose as other factories

S ces nt,

EVENT VALUE

START_ELEMENT = 1
END_ELEMNET = 2

CHARACTERS = 4 (CDATA / text)

COMMENT = 5

START_DOCUMENT = 7
END_DOCUMENT = 8

ATTRIBUTE = 10

NAMESPACE = 13

There is NO any parent / child nodes or methods to retrieve / add them as DOM.

in StAX api, the order of event is the matter. no need to think or worry about hierarchy.

any those logic has to implement in our application only

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

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

abstract Attribute	**createAttribute**(QN Create a new Attribu
abstract Attribute	**createAttribute**(java.lang.String localName, java.lang.String value) Create a new Attribute

None of these CREATION method throws any checked exception.. good

abstract Attribute A	**createAttribute**(java.lang.String prefix, java.lang.String namespaceURI, java.lang.String localName, java.lang.String value) Create a new Attribute
abstract Characters C	**createCData**(java.lang.String content) Create a Characters event with the CData flag set to true No separate class for CDATA !!
abstract Characters C	**createCharacters**(java.lang.String content) Create a Characters event, this method does not check if the content is all whitespace.
abstract Comment C	**createComment**(java.lang.String text) Create a comment
~~abstract DTD~~	~~**createDTD**(java.lang.String dtd)~~ ~~Create a document type definition event This string contains the entire document type declaration that matches the doctypedecl in the XML 1.0 specification~~
abstract EndDocument D	**createEndDocument**() Creates a new instance of an EndDocument event
abstract EndElement E	**createEndElement**(QName name, java.util.Iterator namespaces) Create a new EndElement
abstract EndElement E	**createEndElement**(java.lang.String prefix, java.lang.String namespaceUri, java.lang.String localName) Create a new EndElement
abstract EndElement E	**createEndElement**(java.lang.String prefix, java.lang.String namespaceUri, java.lang.String localName, java.util.Iterator namespaces) Create a new EndElement
~~abstract EntityReference~~	~~**createEntityReference**(java.lang.String name, EntityDeclaration declaration)~~ ~~Creates a new instance of a EntityReference event~~
~~abstract Characters~~	~~**createIgnorableSpace**(java.lang.String content)~~ ~~Create an ignorable space~~
abstract Namespace N	**createNamespace**(java.lang.String namespaceURI) Create a new default Namespace
abstract Namespace N	**createNamespace**(java.lang.String prefix, java.lang.String namespaceUri) Create a new Namespace
~~abstract ProcessingInstruction~~	~~**createProcessingInstruction**(java.lang.String target, java.lang.String data)~~ ~~Create a processing instruction~~
abstract Characters C	**createSpace**(java.lang.String content) Create a Characters event with the isSpace flag set to true

abstract StartDocument	**createStartDocument**() ~~Creates a new instance of a StartDocument event~~
abstract StartDocument	**createStartDocument**(java.lang.String encoding) ~~Creates a new instance of a StartDocument event~~
abstract StartDocument	**createStartDocument**(java.lang.String encoding, java.lang.String version) ~~Creates a new instance of a StartDocument event~~
abstract StartDocument	~~**createStartDocument**(java.lang.String encoding, java.lang.String version, boolean standalone)~~ ~~Creates a new instance of a StartDocument event~~ the correspoding method not in StreamWriter
abstract StartElement	**createStartElement**(QName name, java.util.Iterator attributes, java.util.Iterator namespaces) Create a new StartElement.
abstract StartElement	**createStartElement**(java.lang.String prefix, java.lang.String namespaceUri, java.lang.String localName) Create a new StartElement.
abstract StartElement	**createStartElement**(java.lang.String prefix, java.lang.String namespaceUri, java.lang.String localName, java.util.Iterator attributes, java.util.Iterator namespaces) Create a new StartElement.
abstract StartElement	**createStartElement**(java.lang.String prefix, java.lang.String namespaceUri, java.lang.String localName, java.util.Iterator attributes, java.util.Iterator namespaces, NamespaceContext context) Create a new StartElement.
static XMLEventFactory	**newInstance**() Create a new instance of the factory
static XMLEventFactory	**newInstance**(java.lang.String factoryId, java.lang.ClassLoader classLoader) Create a new instance of the factory
abstract void	**setLocation**(Location location) This method allows setting of the Location on each event that is created by this factory.

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

Constructor Detail

XMLEventFactory

```
protected XMLEventFactory()
```

Method Detail

newInstance

```
public static XMLEventFactory newInstance()
                                    throws FactoryConfigurationError
```

~~Create a new instance of the factory~~

~~**Throws:**~~

`FactoryConfigurationError` ~~- if an instance of this factory cannot be loaded~~

---

newInstance

```
public static XMLEventFactory newInstance(java.lang.String factoryId,
                                          java.lang.ClassLoader classLoader)
                                    throws FactoryConfigurationError
```

~~Create a new instance of the factory~~

```
System.setProperty("jaxp.debug", "1");

 call this to see what is happening while creating
factory instances.
```

~~**Parameters:**~~

`factoryId` ~~- Name of the factory to find, same as a property name~~

`classLoader` ~~- classLoader to use~~

~~**Returns:**~~

~~the factory implementation~~

~~**Throws:**~~

`FactoryConfigurationError` ~~- if an instance of this factory cannot be loaded~~

How can i create object to this Location Either XMLOutputFactory or XMLEventWriter dont have method to get Location ??

i hope this is a GAP in their design (it is not design GAP. it comes only if it is read from XML file but not by on memory creation)

setLocation

```
public abstract void setLocation(Location
```

This method allows setting of the Location on each event that is created by this factory. The values are copied by value into the events created by this factory. To reset the location information set the location to null.

i dont know, purpose of this method. why need to set location object ?

ANS :: actually, we creating event object which means we are creating xml content on memory. to set location information will help to each event any debug BEFORE to go hard disk. Even EventWriter has been associated with a file location, it is will associated with event object only when writing to hard disk. if need in between, we have to set Location object

createAttribute

```
public abstract Attribute createAttribute(java.lang.String prefix,
                                          java.lang.String namespaceURI,
                                          java.lang.String localName,
                                          java.lang.String value)
```

~~Create a new Attribute~~

~~Parameters:~~

`prefix` ~~- the prefix of this attribute, may not be null~~

`namespaceURI` ~~- the attribute value is set to this value, may not be null~~

`localName` ~~- the local name of the XML name of the attribute, localName cannot be null~~

`value` ~~- the attribute value to set, may not be null~~

~~Returns:~~

~~the Attribute with specified values~~

---

createAttribute

```
public abstract Attribute createAttribute(java.lang.String localName,
                                          java.lang.String value)
```

~~Create a new Attribute~~

~~Parameters:~~

`localName` ~~- the local name of the XML name of the attribute, localName cannot be null~~

`value` ~~- the attribute value to set, may not be null~~

~~Returns:~~

~~the Attribute with specified values~~

---

createAttribute

```
public abstract Attribute createAttribute(QName name,
                                          java.lang.String value)
```

~~Create a new Attribute~~

~~Parameters:~~

`name` ~~- the qualified name of the attribute, may not be null~~

`value` ~~- the attribute value to set, may not be null~~

~~Returns:~~

~~the Attribute with specified values~~

---

createNamespace

```
public abstract Namespace createNamespace(java.lang.String namespaceURI)
```

~~Create a new default Namespace~~

it will have defaultNamespace prefix that is empty space

~~**Parameters:**~~

~~`namespaceURI` - the default namespace uri~~

~~**Returns:**~~

~~the Namespace with the specified value~~

---

createNamespace

```
public abstract Namespace createNamespace(java.lang.String prefix,
                                          java.lang.String namespaceUri)
```

~~Create a new Namespace~~

~~**Parameters:**~~

~~`prefix` - the prefix of this namespace, may not be null~~

~~`namespaceUri` - the attribute value is set to this value, may not be null~~

~~**Returns:**~~

~~the Namespace with the specified values~~

KEEP IN MIND, iterator cannot be re used. once used just throw out and create another one param !!!!!

---

createStartElement

iterator of Collection which has one or more Attribute event objects

```
public abstract StartElement createStartElement(QName name,
                                                java.util.Iterator attributes,
                                                java.util.Iterator namespaces)
```

Create a new StartElement. Namespaces can be added to this StartElement by passing in an Iterator that walks over a set of Namespace interfaces. Attributes can be added to this StartElement by passing an iterator that walks over a set of Attribute interfaces.

iterator of collection which has one or more Namespace event object

Parameters:

`name` - the qualified name of the attribute, may not be null

`attributes` - an optional unordered set of objects that implement Attribute to add to the new StartElement, may be null

`namespaces` - an optional unordered set of objects that implement Namespace to add to the new StartElement, may be null

Returns:

an instance of the requested StartElement

---

createStartElement

Every StartElement will have its own NamespaceContext

```
public abstract StartElement createStartElement(java.lang.String prefix,
                                                java.lang.String namespaceUri,
                                                java.lang.String localName)
```

Create a new StartElement. This defaults the NamespaceContext to an empty NamespaceContext. Querying this event for its namespaces or attributes will result in an empty iterator being returned.

mostly, we use this method often, to create start element

Parameters:

`namespaceUri` - the uri of the QName of the new StartElement
`localName` - the local name of the QName of the new StartElement
`prefix` - the prefix of the QName of the new StartElement

Returns:

an instance of the requested StartElement

---

createStartElement

KEEP IN MIND, iterator cannot be re used. once used just throw out and create another one param !!!!!

```
public abstract StartElement createStartElement(java.lang.String prefix,
                                                java.lang.String namespaceUri,
                                                java.lang.String localName,
                                                java.util.Iterator attributes,
                                                java.util.Iterator namespaces)
```

i hope, namespaceUri and namesapaces dont have any difference. but both parameter will have simply namespaces

why

~~Create a new StartElement. Namespaces can be added to this StartElement by passing in an Iterator that walks over a set of Namespace interfaces. Attributes can be added to this StartElement by passing an iterator that walks over a set of Attribute interfaces.~~

Parameters:

`namespaceUri` - the uri of the QName of the new StartElement
`localName` - the local name of the QName of the new StartElement
`prefix` - the prefix of the QName of the new StartElement
`attributes` - an unordered set of objects that implement Attribute to add to the new StartElement
`namespaces` - an unordered set of objects that implement Namespace to add to the new StartElement

Returns:

an instance of the requested StartElement

---

createStartElement

```
public abstract StartElement createStartElement(java.lang.String prefix,
```

what is the purpose of namespacecontext ??

it is just container for a EVERY StartElement event ,which will hold of list of Namespace

```
java.lang.String namespaceUri,
java.lang.String localName,
java.util.Iterator attributes,
java.util.Iterator namespaces,
NamespaceContext context)
```

~~Create a new StartElement. Namespaces can be added to t~~ ~~valks over a set of Namespace interfaces. Attributes can be added to th~~ ~~s over a set of Attribute interfaces.~~

How can i create this NamespaceContext object ??

i hope this is 2nd GAP in their design

Parameters:

`namespaceUri` - the uri of the QName of the new StartElement

`localName` - the local name of the QName of the new StartElement

`prefix` - the prefix of the QName of the new StartElement

`attributes` - an unordered set of objects that implement Attribute to add to the new StartElement, may be null

`namespaces` - an unordered set of objects that implement Namespace to add to the new StartElement, may be null

`context` - the namespace context of this element

Returns:

an instance of the requested StartElement

---

createEndElement

```
public abstract EndElement createEndElement(QName name,
                                            java.util.Iterator namespaces)
```

Create a new EndElement

2 - parameters

Attribute never be with end element

Parameters:

`name` - the qualified name of the EndElement

`namespaces` - an optional unordered set of objects that implement Namespace that have gone out of scope, may be null

Returns:

an instance of the requested EndElement

---

createEndElement

```
public abstract EndElement createEndElement(java.lang.String prefix,
                                            java.lang.String namespaceUri,
                                            java.lang.String localName)
```

Create a new EndElement

3 - parameters

Parameters:

`namespaceUri` - the uri of the QName of the new StartElement

`localName` - the local name of the QName of the new StartElement

`prefix` - the prefix of the QName of the new StartElement

Returns:

an instance of the requested EndElement

---

createEndElement

```
public abstract EndElement createEndElement(java.lang.String prefix,
                                            java.lang.String namespaceUri,
                                            java.lang.String localName,
                                            java.util.Iterator namespaces)
```

4 - parameters

Create a new EndElement

Parameters:

`namespaceUri` - the uri of the QName of the new StartElement

`localName` - the local name of the QName of the new StartElement

`prefix` - the prefix of the QName of the new StartElement

`namespaces` - an unordered set of objects that implement Namespace that have gone out of scope, may be null

Returns:

an instance of the requested EndElement

---

createCharacters

```
public abstract Characters createCharacters(java.lang.String content)
```

Create a Characters event, this method does not check if the content is all whitespace. To create a space event use #createSpace(String)

Parameters:

`content` - the string to create

Returns:

a Characters event

---

createCData

```
public abstract Characters createCData(java.lang.String content)
```

Create a Characters event with the CData flag set to true

Parameters:
`content` - the string to create
Returns:
a Characters event

Really super style of design of Characters class with set of flags

---

createSpace

`public abstract Characters createSpace(java.lang.String content)`

Create a Characters event with the isSpace flag set to true

Parameters:
`content` - the content of the space to create
Returns:
a Characters event

~~createIgnorableSpace~~

~~`public abstract Characters createIgnorableSpace(java.lang.String content)`~~

~~Create an ignorable space~~

~~**Parameters:**~~
~~`content` - the space to create~~
~~**Returns:**~~
~~a Characters event~~

for DTD

---

createStartDocument

`public abstract StartDocument createStartDocument()`

~~Creates a new instance of a StartDocument event~~

use this method. it is enough for real coding

~~**Returns:**~~
~~a StartDocument event~~

---

createStartDocument

`public abstract StartDocument createStartDocument(java.lang.String encoding,`

```
                                                java.lang.String version,
                                                boolean standalone)
```

Creates a new instance of a StartDocument event

standalone, Doesnot appear even if it set to TRUE

it is DTD related

value of these can be any randomly value.
not done any validation.

Parameters:

`encoding` - the encoding style
`version` - the XML version
`standalone` - the status of standalone may be set to "true" or "false"

Returns:

a StartDocument event

---

createStartDocument

```
public abstract StartDocument createStartDocument(java.lang.String encoding,
                                                  java.lang.String version)
```

~~Creates a new instance of a StartDocument event~~

encoding and version can be any arbitrary value. (i tested)

~~**Parameters:**~~

~~`encoding` - the encoding style~~
~~`version` - the XML version~~

~~**Returns:**~~

~~a StartDocument event~~

---

createStartDocument

```
public abstract StartDocument createStartDocument(java.lang.String encoding)
```

Creates a new instance of a StartDocument event

this encoding value can be anything. (i tested)

Parameters:

`encoding` - the encoding style

Returns:

a StartDocument event

---

createEndDocument

```
public abstract EndDocument createEndDocument()
```

~~Creates a new instance of an EndDocument event~~

Returns:
an EndDocument event

---

createEntityReference

```
public abstract EntityReference createEntityReference(java.lang.String name,
                                                      EntityDeclaration declaration)
```

Creates a new instance of a EntityReference event

Parameters:
`name` - The name of the reference
`declaration` - the declaration for the event

Returns:
an EntityReference event

---

createComment

```
public abstract Comment createComment(java.lang.String text)
```

Create a comment

Parameters:
`text` - The text of the comment a Comment event

---

createProcessingInstruction

```
public abstract ProcessingInstruction createProcessingInstruction(java.lang.String target,
                                                                  java.lang.String data)
```

Create a processing instruction

Parameters:
`target` - The target of the processing instruction
`data` - The text of the processing instruction

Returns:
a ProcessingInstruction event

~~createDTD~~

```
public abstract DTD createDTD(java.lang.String dtd)
```

~~Create a document type definition event This string contains the entire document type declaration that matches the doctypedecl in the XML 1.0 specification~~

~~Parameters:~~

`dtd` ~~- the text of the document type definition~~

~~Returns:~~

~~a DTD event~~

StAX API dont have any special method to call for parsing..... super design ..since client control the thread

21 - Nov - 08

31 - Dec - 08

7 methods

java.xml.stream

Interface XMLEventReader

Before to study this class first finish, XMLEventFactory and its individual Events

All Superinterfaces:

java.util.Iterator

All Known Implementing Classes:

EventReaderDelegate

if Stream / Event reader dont have handy method to check event types , we can only check by comparing current event type with XMLStreamConstants

---

`public interface` **`XMLEventReader`**

extends java.util.Iterator

This is the top level interface for parsing XML Events. It provides the ability to peek at the next event and returns configuration information through the property interface.

Since:

1.6

Version:

1.0

Author:

See Also:

`XMLInputFactory`, `XMLEventWriter`

Method Summary

`void`	**`close`**`()` ~~Frees any resources associated with this Reader~~.

which method can use? it dont have hasText() method as cursorAPI

java.lang.String	**getElementText**() Reads the content of a text-only element.
java.lang.Object	**getProperty**(java.lang.String name) Get the value of a feature/property from the underlying implementation
boolean	**hasNext**() Check if there are more events.
XMLEvent	**nextEvent**() Get the next XMLEvent
XMLEvent	**nextTag**() Skips any insignificant space events until a START_ELEMENT or END_ELEMENT is reached.
XMLEvent	**peek**() Check the next XMLEvent without reading it from the stream.

Avoid to use this method

NO Set mtd

these 3 method enough

it is same as cursor API

Avoid to use this method

try to avoid this method also to avoid confusing coding..

Methods inherited fr ... ator
next, remove

Method Detail

nextEvent

```
XMLEvent nextEvent()
                    throws XMLStreamException
```

Get the next XMLEvent

Throws:

XMLStreamException - if there is an error with the underlying XML.

NoSuchElementException - iteration has no more elements.

See Also:

XMLEvent

hasNext

```
boolean hasNext()
```

Check if there are more events. Returns true if there are more events and false otherwise.

Specified by:
`hasNext` in interface `java.util.Iterator`

~~**Returns:**~~
~~true if the event reader has more events, false otherwise~~

---

peek

this is without moving cursor
get the forth coming event.

```
XMLEvent peek()
           throws XMLStreamException
```

Check the next XMLEvent without reading it from the stream. Returns null if the stream is at EOF or has no more XMLEvents. A call to peek() will be equal to the next return of next().

Throws:
`XMLStreamException`

See Also:
`XMLEvent`

---

so, it has to called only a element, it has only TEXT but not any child element.

i dont know, what method i have to check to avoid getting XMLStreamException

getElementText

```
java.lang.String getElementText()
                 throws XMLStreamException
```

Reads the content of a text-only element. Precondition: the current event is START_ELEMENT. Postcondition: The current event is the corresponding END_ELEMENT.

Throws:
`XMLStreamException` - if the current event is not a START_ELEMENT or if a non text element is encountered

---

nextTag

Please read my comment in XMLStreamReader.nextTag() method to understand about this method.

```
XMLEvent nextTag()
                throws XMLStreamException
```

Skips any insignificant space events until a START_ELEMENT or END_ELEMENT is reached. If anything other than space characters are encountered, an exception is thrown. This method should be used when processing element-only content because the parser is not able to recognize ignorable whitespace if the DTD is missing or not interpreted.

Throws:

`XMLStreamException` - if anything other than space characters are encountered

---

getProperty

```
java.lang.Object getProperty(java.lang.String name)
                             throws java.lang.IllegalArgumentException
```

~~Get the value of a feature/property from the underlying implementation~~

~~**Parameters:**~~

`name` ~~- The name of the property~~

~~**Returns:**~~

~~The value of the property~~

Throws:

`java.lang.IllegalArgumentException` - if the property is not supported

---

close

```
void close()
          throws XMLStreamException
```

Frees any resources associated with this Reader. This method does not close the underlying input source.

Throws:

`XMLStreamException` - if there are errors freeing associated resources

---

31 - Dec -08

9 methods

5 - namespace related
2 - add xml events
2 - writer related

javax.xml.stream

Interface XMLEventWriter

All Superinterfaces:

XMLEventConsumer

Before to study this class first finish, XMLEventFactory and its individual Events

if Stream / Event reader dont have handy method to check event types , we can only check by comparing current event type with XMLStreamConstants

we never worry about type of event. since it is writer....

in 5th round, get clear idea about using namespace, prefix methods by getting sample from Net

`public`

extends XI

This is the documents. Instances of this interface are not required to validate the form of the XML.

Since:

1.6

Version:

1.0

Author:

Copyright (c) 2003 by BEA Systems. All Rights Reserved.

See Also:

`XMLEventReader`, `XMLEvent`, `Characters`, `ProcessingInstruction`, `StartElement`, `EndElement`

inherit from super interface

add one by one

Method Summary

void	**add**(XMLEvent event) Add an event to the output stream Adding a START_ELEMENT will open a new namespace scope that will be closed when the corresponding END_ELEMENT is written.
void	**add**(XMLEventReader reader) Adds an entire stream to an output stream, calls next() on the inputStream argument until hasNext() returns false This should be treated as a convenience method that will perform the following loop over all the events in an event reader and call add on each event.
void	**close**() ~~Frees any resources associated with this stream~~

parameter event reader only, but not stream reader

add as bulk

good for simply copy and paste of XML file

void	**flush**() ~~Writes any cached events to the underlying output mechanism~~
NamespaceContext	**getNamespaceContext**() ~~Returns the current namespace context.~~
java.lang.String	**getPrefix**(java.lang.String uri) ~~Gets the prefix the uri is bound to~~
void	**setDefaultNamespace**(java.lang.String uri) Binds a URI to the default namespace This URI is bound in the scope of the current START_ELEMENT / END_ELEMENT pair.
void	**setNamespaceContext**(NamespaceContext context) ~~Sets the current namespace context for prefix and uri bindings.~~
void	**setPrefix**(java.lang.String prefix, java.lang.String uri) ~~Sets the prefix the uri is bound to.~~

all of these 5 methods are namespace related..

Method Detail

flush

```
void flush()
        throws XMLStreamException
```

~~Writes any cached events to the underlying output mechanism~~

Throws:
XMLStreamException

---

close

```
void close()
        throws XMLStreamException
```

~~Frees any resources associated with this stream~~

Throws:
XMLStreamException

---

add

```
void add(XMLEvent event)
         throws XMLStreamException
```

Add an event to the output stream. Adding a START_ELEMENT will open a new namespace scope that will be closed when the corresponding END_ELEMENT is written.

Required and optional fields for events added to the writer

Event Type	Required Fields	Optional Fields	Required Behavior
START_ELEMENT	QName name	namespaces , attributes	A START_ELEMENT will be written by writing the name, namespaces, and attributes of the event in XML 1.0 valid syntax for START_ELEMENTs. The name is written by looking up the prefix for the namespace uri. The writer can be configured to respect prefixes of QNames. If the writer is respecting prefixes it must use the prefix set on the QName. The default behavior is to lookup the value for the prefix on the EventWriter's internal namespace context. Each attribute (if any) is written using the behavior specified in the attribute section of this table. Each namespace (if any) is written using the behavior specified in the namespace section of this table.

so, get prefix from QName and look its uri from namespacecontext object.

END_ELEMENT	Qname name	None	A well formed END_ELEMENT tag is written. The name is written by looking up the prefix for the namespace uri. The writer can be configured to respect prefixes of QNames. If the writer is respecting prefixes it must use the prefix set on the QName. The default behavior is to lookup the value for the prefix on the EventWriter's internal namespace context. If the END_ELEMENT name does not match the START_ELEMENT name an XMLStreamException is thrown.
ATTRIBUTE	QName name , String value	QName type	An attribute is written using the same algorithm to find the lexical form as used in START_ELEMENT. The default is to use double quotes to wrap attribute values and to escape any double quotes found in the value. The type value is ignored.
NAMESPACE	String prefix, String namespaceURI, boolean isDefaultNamespaceDeclaration	None	A namespace declaration is written. If the namespace is a default namespace declaration (isDefault is true) then xmlns="$namespaceURI" is written and the prefix is optional. If isDefault is false, the prefix must be declared and the writer must prepend xmlns to the prefix and write out a standard prefix declaration.

~~PROCESSING_INSTRUCTION~~	~~None~~	~~String target, String data~~	~~The data does not need to be present and may be null. Target is required and many not be null. The writer will write data section directly after the target, enclosed in appropriate XML 1.0 syntax~~
COMMENT	None	String comment	If the comment is present (not null) it is written, otherwise an an empty comment is written
START_DOCUMENT	None wow.. it is optional	String encoding , boolean standalone, String version	A START_DOCUMENT event is not required to be written to the stream. If present the attributes are written inside the appropriate XML declaration syntax
END_DOCUMENT	None	None	Nothing is written to the output
~~DTD~~	~~String DocumentTypeDefinition~~	~~None~~	~~The DocumentTypeDefinition is written to the output~~

Specified by:

add in interface XMLEventConsumer

Parameters:

`event` - the event to be added

Throws:

XMLStreamException

---

add

```
void add(XMLEventReader reader)
         throws XMLStreamException
```

~~Adds an entire stream to an output stream, calls next() on the inputStream argument until hasNext() returns false This should be treated as a convenience method that will perform the following loop over all the events in an event reader and call add on each event.~~

Parameters:

~~`reader` - the event stream to add to the output~~

~~**Throws:**~~

`XMLStreamException`

---

getPrefix

```
java.lang.String getPrefix(java.lang.String uri)
                           throws XMLStreamException
```

Gets the prefix the uri is bound to

Parameters:

`uri` - the uri to look up

Throws:

`XMLStreamException`

---

setPrefix

```
void setPrefix(java.lang.String prefix,
               java.lang.String uri)
               throws XMLStreamException
```

prefix is not added to output xml. i tested.

get sample in 5th round

Sets the prefix the uri is bound to. This prefix is bound in the scope of the current START_ELEMENT / END_ELEMENT pair. If this method is called before a START_ELEMENT has been written the prefix is bound in the root scope.

Parameters:

`prefix` - the prefix to bind to the uri

`uri` - the uri to bind to the prefix

Throws:

`XMLStreamException`

---

setDefaultNamespace

```
void setDefaultNamespace(java.lang.String uri)
                         throws XMLStreamException
```

Binds a URI to the default namespace This URI is bound in the scope of the current START_ELEMENT / END_ELEMENT pair. If this method is called before a START_ELEMENT has been written the uri is bound in the root scope.

Parameters:
`uri` - the uri to bind to the default namespace

Throws:
`XMLStreamException`

---

setNamespaceContext

```
void setNamespaceContext(NamespaceContext context)
                         throws XMLStreamException
```

Sets the current namespace context for prefix and uri bindings. This context becomes the root namespace context for writing and will replace the current root namespace context. Subsequent calls to setPrefix and setDefaultNamespace will bind namespaces using the context passed to the method as the root context for resolving namespaces.

Parameters:
`context` - the namespace context to use for this writer

Throws:
`XMLStreamException`

---

getNamespaceContext

```
NamespaceContext getNamespaceContext()
```

~~Returns the current namespace context.~~

~~Returns:~~
~~the current namespace context~~

---

15 - Nov - 08

26 - Dec - 08

25 methods

2 - instance method
2 - filtered reader
7 - overloaded EventReader
6 - overloaded StreamReader
8 - getter / setter methods

jav

Class XMLInputFactory

```
java.lang.Object
  └javax.xml.stream.XMLInputFactory
```

```
public abstract class
```

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

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

extends java.lang.Object

Defines an abstract implementati tandard properties of this specification. Each property varies in the level of support required by each implementation. The level of support required

At 5th round, be clear idea about QName, NamespaceContext. Namespace(it is StAXEvent) these are used in StAX api at some great Extend

Configuration parameters

Property Name	Behavior	Return type	Default Value	Required
javax.xml.stream.isValidating	Turns on/off implementation specific DTD validation	Boolean	False	No
javax.xml.stream. isNamespaceAware	Turns on/off namespace processing for XML 1.0 support	Boole Turn off this one to improve performance of processor	True	True (required) / False (optional)
javax.xml.stream.is			False	Yes
~~javax.xml.stream. isReplacingEntityR~~	~~characters~~		~~True~~	~~Yes~~
~~javax.xml.~~stream. ~~isSupportingExternalEntities~~	~~Resolve external parsed entities~~	~~Boolean~~	~~Unspecified~~	~~Yes~~

There is NO any parent / child nodes or methods to retrieve / add them as DOM. thats why stax has StartElement and EndElement instead of simply Element

in StAX api, the order of event is the matter. no need to think or worry about hierarchy.

any those logic has to implement in our application only

set to FALSE

javax.xml.stream.supportDTD	Use this property to request processors that do not support DTDs	Boolean	True	Yes
javax.xml.stream.reporter	sets/gets the impl of the XMLReporter	javax.xml.stream.XMLReporter	Null	Yes
javax.xml.stream.resolver	sets/gets the impl of the XMLResolver interface	javax.xml.stream.XMLResolver	Null	Yes
javax.xml.stream.allocator	sets/gets the impl of the XMLEventAllocator interface	javax.xml.stream.util.XMLEventAllocator	Null	Yes

Since:
1.6
Version:
1.0
Author:

See Also:
`XMLOutputFactory`, `XMLEventReader`, `XMLStreamReader`, `EventFilter`, `XMLReporter`, `XMLResolver`, `XMLEventAllocator`

Field Summary

`static java.lang.String`	**`ALLOCATOR`** The property used to set/get the implementation of the allocator
`static java.lang.String`	**`IS_COALESCING`** The property that requires the parser to coalesce adjacent character data sections
`static java.lang.String`	**`IS_NAMESPACE_AWARE`** The property used to turn on/off namespace support, this is to support XML 1.0 documents, only the true setting must be supported
~~`static java.lang.String`~~	~~**`IS_REPLACING_ENTITY_REFERENCES`**~~ ~~Requires the parser to replace internal entity references with their replacement text and report them as characters~~
~~`static java.lang.String`~~	~~**`IS_SUPPORTING_EXTERNAL_ENTITIES`**~~ ~~The property that requires the parser to resolve external parsed entities~~
~~`static java.lang.String`~~	~~**`IS_VALIDATING`**~~ ~~The property used to turn on/off implementation specific validation~~

static java.lang.String	**REPORTER** The property us ... ace
static java.lang.String	**RESOLVER** The property us
static java.lang.String	**SUPPORT_DTD** The property tha

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

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

Constructor Summary

protected	**XMLInputFactory**()

source for stream reader can be either one among THREE
1. Source (trax)
2. InputStream
3. Reader

Method Summary

abstract XMLEventReader	**createFilteredReader**(XMLEventReader reader, EventFilter filter) Create a filtered event reader that wraps the filter around the event reader
abstract XMLStreamReader	**createFilteredReader**(XMLStreamReader reader, StreamFilter filter) Create a filtered reader that wraps the filter around the reader
abstract XMLEventReader	**createXMLEventReader**(java.io.InputStream stream) Create a new XMLEventReader from a java.io.InputStream
abstract XMLEventReader	**createXMLEventReader**(java.io.InputStream stream, java.lang.String encoding) Create a new XMLEventReader from a java.io.InputStream
abstract XMLEventReader	**createXMLEventReader**(java.io.Reader reader) Create a new XMLEventReader from a reader
abstract XMLEventReader	**createXMLEventReader**(Source source) Create a new XMLEventReader from a JAXP source.
abstract XMLEventReader	**createXMLEventReader**(java.lang.String systemId, java.io.InputStream stream) Create a new XMLEventReader from a java.io.InputStream
abstract XMLEventReader	**createXMLEventReader**(java.lang.String systemId, java.io.Reader reader) Create a new XMLEventReader from a reader
abstract XMLEventReader	**createXMLEventReader**(XMLStreamReader reader) Create a new XMLEventReader from an XMLStreamReader.
abstract XMLStreamReader	**createXMLStreamReader**(java.io.InputStream stream) Create a new XMLStreamReader from a java.io.InputStream

abstract XMLStreamReader	**createXMLStreamReader**(java.io.InputStream stream, java.lang.String encoding) Create a new XMLStreamReader from a java.io.InputStream
abstract XMLStreamReader	**createXMLStreamReader**(java.io.Reader reader) Create a new XMLStreamReader from a reader
abstract XMLStreamReader	**createXMLStreamReader**(Source source) Create a new XMLStreamReader from a JAXP source.
abstract XMLStreamReader	**createXMLStreamReader**(java.lang.String systemId, java.io.InputStream stream) Create a new XMLStreamReader from a java.io.InputStream
abstract XMLStreamReader	**createXMLStreamReader**(java.lang.String systemId, java.io.Reader reader) Create a new XMLStreamReader from a java.io.InputStream
abstract XMLEventAllocator	**getEventAllocator**() Gets the allocator used by streams created with this factory
abstract java.lang.Object	**getProperty**(java.lang.String name) ~~Get the value of a feature/property from the underlying implementation~~
abstract XMLReporter	**getXMLReporter**() ~~The reporter that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~
abstract XMLResolver	~~**getXMLResolver**()~~ ~~The resolver that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~
abstract boolean	**isPropertySupported**(java.lang.String name) ~~Query the set of properties that this factory supports~~.
static XMLInputFactory	**newInstance**() ~~Create a new instance of the factory~~.
static XMLInputFactory	**newInstance**(java.lang.String factoryId, java.lang.ClassLoader classLoader) ~~Create a new instance of the factory~~
abstract void	**setEventAllocator**(XMLEventAllocator allocator) Set a user defined event allocator for events
abstract void	**setProperty**(java.lang.String name, java.lang.Object value) ~~Allows the user to set specific feature/property on the underlying implementation.~~
abstract void	**setXMLReporter**(XMLReporter reporter) ~~The reporter that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~

3 method with inputStream

use this method as best practice

2 method with Reader

~~abstract void~~	~~**setXMLResolver**(XMLResolver resolver)~~ ~~The resolver that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~

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

Field Detail

IS_NAMESPACE_AWARE

public static final java.lang.String **IS_NAMESPACE_AWARE**

~~The property used to turn on/off namespace support, this is to support XML 1.0 documents, only the true setting must be supported~~

~~**See Also:**~~
Constant Field Values

IS_VALIDATING

public static final java.lang.String **IS_VALIDATING**

~~The property used to turn on/off implementation specific validation~~

~~**See Also:**~~
~~Constant Field Values~~

IS_COALESCING

these variable will have just KEY

public static final java.lang.String **IS_COALESCING**

~~The property that requires the parser to coalesce adjacent character data sections~~

~~**See Also:**~~
~~Constant Field Values~~

~~IS_REPLACING_ENTITY_REFERENCES~~

~~`public static final java.lang.String` **`IS_REPLACING_ENTITY_REFERENCES`**~~

~~Requires the parser to replace internal entity references with their replacement text and report them as characters~~

~~**See Also:**~~
~~Constant Field Values~~

~~IS_SUPPORTING_EXTERNAL_ENTITIES~~

~~`public static final java.lang.String` **`IS_SUPPORTING_EXTERNAL_ENTITIES`**~~

~~The property that requires the parser to resolve external parsed entities~~

~~**See Also:**~~
~~Constant Field Values~~

~~SUPPORT_DTD~~

~~`public static final java.lang.String` **`SUPPORT_DTD`**~~

~~The property that requires the parser to support DTDs~~

~~**See Also:**~~
~~Constant Field Values~~

~~REPORTER~~

~~`public static final java.lang.String` **`REPORTER`**~~

~~The property used to set/get the implementation of the XMLReporter interface~~

~~**See Also:**~~
~~Constant Field Values~~

RESOLVER

~~public static final java.lang.String **RESOLVER**~~

~~The property used to set/get the implementation of the XMLResolver~~

~~**See Also:**~~
~~Constant Field Values~~

~~**ALLOCATOR**~~

~~public static final java.lang.String **ALLOCATOR**~~

~~The property used to set/get the implementation of the allocator~~

~~**See Also:**~~
~~Constant Field Values~~

Constructor Detail

XMLInputFactory

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

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

```
protected XMLInputFactory()
```

Method Detail

newInstance

for, ERROR, no need to put any try-catch AS for checked exception

```
public static XMLInputFactory newInstance()
                                throws FactoryConfigurationError
```

~~Create a new instance of the factory. This static method creates a new factory instance. This method uses the following ordered lookup procedure to determine the XMLInputFactory implementation class to load: Use the javax.xml.stream.XMLInputFactory system property. Use the properties file "lib/stax.properties" in the JRE directory. This configuration file is in standard java.util.Properties format and contains the fully qualified name of the implementation class with the key being the system property defined above. Use the Services API (as detailed in the JAR specification), if available, to determine the classname. The Services API will look for a classname in the file META-INF/services/javax.xml.stream.XMLInputFactory in jars available to the runtime. Platform default XMLInputFactory instance. Once an application has obtained a reference to a XMLInputFactory it can use the factory to configure and obtain stream instances.~~

Throws:
`FactoryConfigurationError` - if an instance of this factory cannot be loaded

---

newInstance

```
public static XMLInputFactory newInstance(java.lang.String factoryId,
                                          java.ClassLoader classLoader)
                                   throws FactoryConfigurationError
```

~~Create a new instance of the factory~~

~~Parameters:~~

`factoryId` ~~- Name of the factory to find, same as a property name~~

`classLoader` ~~- classLoader to use~~

~~Returns:~~

~~the factory implementation~~

~~Throws:~~

`FactoryConfigurationError` ~~- if an instance of this factory cannot be loaded~~

---

createXMLStreamReader

```
public abstract XMLStreamReader createXMLStreamReader(java.io.Reader reader)
                                                throws XMLStreamException
```

~~Create a new XMLStreamReader from a reader~~

~~Parameters:~~

`reader` ~~- the XML data to read from~~

~~Throws:~~

`XMLStreamException`

---

createXMLStreamReader

this is not InputSoruce used in SAX.
it is comes from TrAX

```
public abstract XMLStreamReader createXMLStreamReader(Source source)
                                                throws XMLStreamException
```

~~Create a new XMLStreamReader from a JAXP source. This method is optional.~~

~~Parameters:~~

`source` ~~- the source to read from~~

~~Throws:~~

`java.lang.UnsupportedOperationException` - if this method is not supported by this

XMLInputFactory
XMLStreamException

---

createXMLStreamReader

```
public abstract XMLStreamReader createXMLStreamReader(java.io.InputStream stream)
                                                      throws XMLStreamException
```

~~Create a new XMLStreamReader from a java.io.InputStream~~

~~Parameters:~~

~~stream - the InputStream to read from~~

~~Throws:~~

~~XMLStreamException~~

---

createXMLStreamReader

```
public abstract XMLStreamReader createXMLStreamReader(java.io.InputStream stream,
                                                      java.lang.String encoding)
                                                      throws XMLStreamException
```

~~Create a new XMLStreamReader from a java.io.InputStream~~

~~Parameters:~~

~~stream - the InputStream to read from~~

~~encoding - the character encoding of the stream~~

~~Throws:~~

~~XMLStreamException~~

this encoding cannot any arbitrary value like "" / "UTF - 9".

it can be UTF-8. i tested

---

createXMLStreamReader

```
public abstract XMLStreamReader createXMLStreamReader(java.lang.String systemId,
                                                      java.io.InputStream stream)
                                                      throws XMLStreamException
```

Create a new XMLStreamReader from a java.io.InputStream

Parameters:

systemId - the system ID of the stream

`stream` - the InputStream to read from

Throws:
`XMLStreamException`

---

createXMLStreamReader

```
public abstract XMLStreamReader createXMLStreamReader(java.lang.String systemId,
                                                      java.io.Reader reader)
                                                 throws XMLStreamException
```

~~Create a new XMLStreamReader from a java.io.InputStream~~

printing mistake

~~**Parameters:**~~
~~`systemId` - the system ID of the stream~~
~~`reader` - the InputStream to read from~~

~~**Throws:**~~
~~`XMLStreamException`~~

duplicate

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(java.io.Reader reader)
                                              throws XMLStreamException
```

~~Create a new XMLEventReader from a reader~~

~~**Parameters:**~~
~~`reader` - the XML data to read from~~

Throws:
`XMLStreamException`

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(java.lang.String systemId,
                                                    java.io.Reader reader)
                                              throws XMLStreamException
```

~~Create a new XMLEventReader from a reader~~

~~**Parameters:**~~

~~`systemId` - the system ID of the input~~
~~`reader` - the XML data to read from~~

~~**Throws:**~~
`XMLStreamException`

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(XMLStreamReader reader)
                                                    throws XMLStreamException
```

Create a new XMLEventReader from an XMLStreamReader. After being used to construct the XMLEventReader instance returned from this method the XMLStreamReader must not be used.

Parameters:
`reader` - the XMLStreamReader to read from (may not be modified)

Returns:
a new XMLEventReader

Throws:
`XMLStreamException`

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(Source source)
                                                    throws XMLStreamException
```

~~Create a new XMLEventReader from a JAXP source. Support of this method is optional.~~

~~**Parameters:**~~
~~`source` - the source to read from~~

Throws:
`java.lang.UnsupportedOperationException` - if this method is not supported by this XMLInputFactory
`XMLStreamException`

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(java.io.InputStream stream)
                                                    throws XMLStreamException
```

Create a new XMLEventReader from a java.io.InputStream

Parameters:
`stream` - the InputStream to read from

Throws:
`XMLStreamException`

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(java.io.InputStream stream,
                                                    java.lang.String encoding)
                                             throws XMLStreamException
```

~~Create a new XMLEventReader from a java.io.InputStream~~

First String parameter is SystemID

Second String parameter is Encoding

~~**Parameters:**~~
~~`stream` - the InputStream to read from~~
~~`encoding` - the character encoding of the stream~~

~~**Throws:**~~
~~`XMLStreamException`~~

---

createXMLEventReader

```
public abstract XMLEventReader createXMLEventReader(java.lang.String systemId,
                                                    java.io.InputStream stream)
                                             throws XMLStreamException
```

~~Create a new XMLEventReader from a java.io.InputStream~~

~~**Parameters:**~~
~~`systemId` - the system ID of the stream~~
~~`stream` - the InputStream to read from~~

~~**Throws:**~~
~~`XMLStreamException`~~

~~createFilteredReader~~

```
public abstract XMLStreamReader createFilteredReader(XMLStreamReader reader,
                                                     StreamFilter filter)
                                              throws XMLStreamException
```

~~Create a filtered reader that wraps the filter around the reader~~

~~Parameters:~~

`reader` ~~- the reader to filter~~

`filter` ~~- the filter to apply to the reader~~

~~Throws:~~

`XMLStreamException`

---

createFilteredReader

```
public abstract XMLEventReader createFilteredReader(XMLEventReader reader,
                                                    EventFilter filter)
                                                 throws XMLStreamException
```

~~Create a filtered event reader that wraps the filter around the event reader~~

~~Parameters:~~

`reader` ~~- the event reader to wrap~~

`filter` ~~- the filter to apply to the event reader~~

~~Throws:~~

`XMLStreamException`

~~getXMLResolver~~

DTD related

```
public abstract XMLResolver getXMLResolver()
```

~~The resolver that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~

~~setXMLResolver~~

```
public abstract void setXMLReso
```

DTD related

```
esolver)
```

~~The resolver that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.~~

~~Parameters:~~

`resolver` ~~- the resolver to use to resolve references~~

---

getXMLReporter

```
public abstract XMLReporter getXMLReporter()
```

The reporter that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.

---

setXMLReporter

```
public abstract void setXMLReporter(XMLReporter reporter)
```

The reporter that will be set on any XMLStreamReader or XMLEventReader created by this factory instance.

Parameters:

`reporter` - the resolver to use to report non fatal errors

---

setProperty

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

no separate method for FEATURE

Allows the user to set specific feature/property on the underlying implementation. The underlying implementation is not required to support every setting of every property in the specification and may use IllegalArgumentException to signal that an unsupported property may not be set with the specified value.

~~**Parameters:**~~

`name` ~~- The name of the property (may not be null)~~

`value` ~~- The value of the property~~

~~**Throws:**~~

`java.lang.IllegalArgumentException` ~~- if the property is not supported~~

---

getProperty

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

~~Get the value of a feature/property from the underlying implementation~~

~~**Parameters:**~~

`name` ~~- The name of the property (may not be null)~~

~~**Returns:**~~
~~The value of the property~~
~~**Throws:**~~
`java.lang.IllegalArgumentException` ~~- if the property is not supported~~

---

isPropertySupported

`public abstract boolean` **`isPropertySupported`**`(java.lang.String name)`

~~Query the set of properties that this factory supports.~~

~~**Parameters:**~~
`name` ~~- The name of the property (may not be null)~~
~~**Returns:**~~
~~true if the property is supported and false otherwise~~

---

setEventAllocator

`public abstract void` **`setEventAllocator`**`(XMLEventAllocator allocator)`

~~Set a user defined event allocator for events~~

~~**Parameters:**~~
`allocator` ~~- the user defined allocator~~

---

getEventAllocator

`public abstract XMLEventAllocator` **`getEventAllocator`**`()`

~~Gets the allocator used by streams created with this factory~~

---

destination can be either

1. Writer
2. OutputStream
3. Result (TrAX)

15 - Nov - 08

27 - Dec - 08

javax.xml.stream

C[illegible]utputFactory

13 methods

4 - methods for StreamWriter
4 - methods for EventWriter
3 - methods for property
2 - methods for instance creation

```
java.lang.Object
  └javax.xml.stream.XMLOutputFactory
```

```
public abstract class XMLOutputFactory
```

extends java.lang.Object

the whole documententaion of this class talks about only namespace repairing

~~Defines an abstract implementation of a factory for getting XMLEventWriters and XMLStreamWriters. The following table defines the standard properties of this specification. Each property varies in the level of support required by each implementation. The level of support required is described in the 'Required' column.~~

Configuration parameters				
Property Name	**Behavior**	**Return type**	**Default Value**	**Required**
javax.xml.stream.isRepairingNamespaces	defaults prefixes on the output side	Boolean	False	Yes

~~The following paragraphs describe the namespace and prefix repair algorithm:~~

~~The property can be se~~ ~~'javax.xml.stream.isRepairingName~~

The next one page of this class talks about only Prefix and Namesapce and its property isRepairngNamespaces

~~This property specifie~~ ~~ons. The default value is false.~~

If a writer isRepairingNamespaces it will create a namespace declaration on the current StartElement for any attribute that does not currently have a namespace declaration in scope. If the StartElement has a uri but no prefix specified a prefix will be assigned, if the prefix has not been declared in a parent of the current StartElement it will be declared on the current StartElement. If the defaultNamespace is bound and in scope and the default namespace matches the URI of the attribute or StartElement QName no prefix will be assigned.

If an element or attribute name has a prefix, but is not bound to any namespace URI, then the prefix will be removed during serialization.

If element and/or attribute names in the same start or empty-element tag are bound to different namespace URIs and are using the same prefix then the element or the first occurring attribute retains the original prefix and the following

ok as usual what

attributes have their prefixes replaced with a new prefix that is bound to the namespace URIs of those attributes.

If an element or attribute name uses a prefix that is bound to a different URI than that inherited from the namespace context of the parent of that element and there is no namespace declaration in the context of the current element then such a namespace declaration is added.

If an element or attribute name is bound to a prefix and there is a namespace declaration that binds that prefix to a different URI then that namespace declaration is either removed if the correct mapping is inherited from the parent context of that element, or changed to the namespace URI of the element or attribute using that prefix.

Since:
1.6
Version:
1.0
Author:

See Also:
`XMLInputFactory`, `XMLEventWriter`, `XMLStreamWriter`

Field Summary

static java.lang.String	**IS_REPAIRING_NAMESPACES** Property used to set prefix defa

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

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

Constructor Summary

protected	**XMLOutputFactory**()

Method Summary

abstract XMLEventWriter	**createXMLEventWriter**(java.io.OutputStream stream) ~~Create a new XMLEventWriter that writes to a stream~~
abstract XMLEventWriter	**createXMLEventWriter**(java.io.OutputStream stream, java.lang.String encoding) ~~Create a new XMLEventWriter that writes to a stream~~
abstract XMLEventWriter	**createXMLEventWriter**(Result result) ~~Create a new XMLEventWriter that writes to a JAXP result.~~
abstract XMLEventWriter	**createXMLEventWriter**(java.io.Writer stream) ~~Create a new XMLEventWriter that writes to a writer~~
abstract XMLStreamWriter	**createXMLStreamWriter**(java.io.OutputStream stream) ~~Create a new XMLStreamWriter that writes to a stream~~

4

abstract XMLStreamWriter	**createXMLStreamWriter**(java.io.OutputStream stream, java.lang.String encoding) ~~Create a new XMLStreamWriter that writes to a stream~~
abstract XMLStreamWriter	**createXMLStreamWriter**(Result result) ~~Create a new XMLStreamWriter that writes to a JAXP result~~.
abstract XMLStreamWriter	**createXMLStreamWriter**(java.io.Writer stream) ~~Create a new XMLStreamWriter that writes to a writer~~
abstract java.lang.Object	**getProperty**(java.lang.String name) ~~Get a feature/property on the underlying implementation~~
abstract boolean	**isPropertySupported**(java.lang.String name) Query the set of properties that this factory supports.
static XMLOutputFactory	**newInstance**() Create a new instance of the factory.
static XMLInputFactory	**newInstance**(java.lang.String factoryId, ClassLoader classLoader) Create a new instance of the factory.
abstract void	**setProperty**(java.lang.String name, java.lang.Object value) ~~Allows the user to set specific features/properties on the underlying implementation~~.

it is bug. it has to return XMLOutputFactory Only

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

Field Detail

IS_REPAIRING_NAMESPACES

public static final java.lang.String **IS_REPAIRING_NAMESPACES**

Property used to set prefix defaulting on the output side

See Also:
Constant Field Values

Constructor Detail

XMLOutputFactory

```
protected XMLOutputFactory()
```

Method Detail

newInstance

```
public static XMLOutputFactory newInstance()
                                 throws FactoryConfigurationError
```

~~Create a new instance of the factory~~.

Throws:
FactoryConfigurationError - if an instance of this factory cannot be loaded

---

newInstance

```
public static XMLInputFactory newInstance(java.lang.String factoryId,
                                          java.lang.ClassLoader classLoader)
                                 throws FactoryConfigurationError
```

~~Create a new instance of the factory~~.

very big bug..

it is not printing mistake.
methods itself like that in XMLOutputFactory.

~~**Parameters:**~~
~~factoryId - Name of the factory to find, same as a property name~~
~~classLoader - classLoader to use~~
~~**Returns:**~~
~~the factory implementation~~
Throws:
FactoryConfigurationError - if an instance of this factory cannot be loaded

---

createXMLStreamWriter

```
public abstract XMLStreamWriter createXMLStreamWriter(java.io.Writer stream)
                                 throws XMLStreamException
```

~~Create a new XMLStreamWriter that writes to a writer~~

~~**Parameters:**~~
~~stream - the writer to write to~~
~~**Throws:**~~

XMLStreamException

---

createXMLStreamWriter

```
public abstract XMLStreamWriter createXMLStreamWriter(java.io.OutputStream stream)
                                                      throws XMLStreamException
```

~~Create a new X~~

i count not able to write this intance . i dont know why ?(commented line)

```
xmlStreamWriter = xmlOutputFactory.createXMLStreamWriter(fileWriter);
   //logger.info("stream writer for file Writer "+xmlStreamWriter);

ANS:  the given implementation has not override toString() method. i
faced the same problem in 5th round study too. ( 27 - DEC - 08 )
```

~~Parameters:~~

~~stream~~

Throws:

XMLStreamException

---

createXMLStreamWriter

```
public abstract XMLStreamWriter createXMLStreamWriter(java.io.OutputStream stream,
                                                      java.lang.String encoding)
                                                      throws XMLStreamException
```

~~Create a new XMLStreamWriter that writes to a stream~~

this encoding cannot be EMPTY.
also, it cannot other than UTF-8 or UTF-16

~~Parameters:~~

~~stream - the stream to write to~~

~~encoding - the encoding to use~~

Throws:

XMLStreamException

---

createXMLStreamWriter

```
public abstract XMLStreamWriter createXMLStreamWriter(Result result)
                                                      throws XMLStreamException
```

Create a new XMLStreamWriter that writes to a JAXP result. This method is optional.

current implementation has this feature

Parameters:

`result` - the result to write to

Throws:

`java.lang.UnsupportedOperationException` - if this method is not supported by this XMLOutputFactory

XMLStreamException

---

createXMLEventWriter

toString() has done in XMLEventWriter but not in XMLStreamWriter

```
public abstract XMLEventWriter createXMLEventWriter(Result result)
                                                    throws XMLStreamException
```

Create a new XMLEventWriter that writes to a JAXP result. This method is optional.

Parameters:

`result` - the result to write to

Throws:

`java.lang.UnsupportedOperationException` - if this method is not supported by this XMLOutputFactory

XMLStreamException

---

createXMLEventWriter

```
public abstract XMLEventWriter createXMLEventWriter(java.io.OutputStream stream)
                                                    throws XMLStreamException
```

~~Create a new XMLEventWriter that writes to a stream~~

~~**Parameters:**~~

~~`stream` - the stream to write to~~

Throws:

XMLStreamException

---

createXMLEventWriter

```
public abstract XMLEventWriter createXMLEventWriter(java.io.OutputStream stream,
                                                    java.lang.String encoding)
                                                    throws XMLStreamException
```

~~Create a new XMLEventWriter that writes to a stream~~

~~**Parameters:**~~

~~`stream` - the stream to write to~~

~~`encoding` - the encoding to use~~

Throws:

XMLStreamException

---

createXMLEventWriter

```
public abstract XMLEventWriter createXMLEventWriter(java.io.Writer stream)
                                                    throws XMLStreamException
```

~~Create a new XMLEventWriter that writes to a writer~~

~~Parameters:~~

`stream` ~~- the stream to write to~~

Throws:

XMLStreamException

---

setProperty

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

~~Allows the user to set specific features/properties on the underlying implementation.~~

~~Parameters:~~

`name` ~~- The name of the property~~

`value` ~~- The value of the property~~

Throws:

`java.lang.IllegalArgumentException` - if the property is not supported

---

getProperty

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

~~Get a feature/property on the underlying implementation~~

~~Parameters:~~

`name` ~~- The name of the property~~

~~Returns:~~

~~The value of the property~~

Throws:

`java.lang.IllegalArgumentException` - if the property is not supported

---

isPropertySupported

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

Query the set of properties that this factory supports.

this method would return true. But it does not mean, factory instance has enabled that property

Parameters:

`name` - The name of the property (may not be null)

Returns:

true if the property is supported and false otherwise

---

27 - Dec - 08

Even this can be used for Non-Fatel error like schema validation, we cannot use since Stax dont support Schema validation

So, it is only for DTD.

1 method

javax.xml.stream

Interface XMLReporter

```
public interface XMLReporter
```

This interface is used to report non-fatal errors. Only warnings should be echoed through this interface.

warning mean, it is related to DTD, i think

Since:
1.6

Version:
1.0

Author:
Copyright (c) 2003 b

There is NO any parent / child nodes or methods to retrieve / add them as DOM.

in StAX api, the order of event is the matter. no need to think or worry about hierarchy.

any those logic has to implement in our application only

Method Summary

void	`report(java.lang.String message, java.lang.String errorType, java.lang.Object relatedInformation, Location location)` Report the desired message in an application specific format.

Method Detail

report

```
void report(java.lang.String message,
            java.lang.String errorType,
            java.lang.Object relatedInformation,
            Location location)
```

javax.xml.stream

Interface XMLStreamConstants

All Known Subinterfaces:

Attribute, Characters, Comment, DTD, EndDocument, EndElement, EntityDeclaration, EntityReference, Namespace, NotationDeclaration, ProcessingInstruction, StartDocument, StartElement, XMLEvent, XMLStreamReader

All Known Implementing Classes:

StreamReaderDelegate

`public interface` **`XMLStreamConstants`**

This interface declares the constants used in this API. Numbers in the range 0 to 256 are reserved for the specification, user defined events must use event codes outside that range.

Since:

1.6

Field Summary

`static int`	**ATTRIBUTE** Indicates an event is an attribute
`static int`	**CDATA** Indicates an event is a CDATA section
`static int`	**CHARACTERS** Indicates an event is characters

static int	**COMMENT** Indicates an event is a comment
static int	**DTD** Indicates an event is a DTD
static int	**END_DOCUMENT** Indicates an event is an end document
static int	**END_ELEMENT** Indicates an event is an end element
static int	**ENTITY_DECLARATION** Indicates a Entity Declaration
static int	**ENTITY_REFERENCE** Indicates an event is an entity reference
static int	**NAMESPACE** Indicates the event is a namespace declaration
static int	**NOTATION_DECLARATION** Indicates a Notation
static int	**PROCESSING_INSTRUCTION** Indicates an event is a processing instruction
static int	**SPACE** The characters are white space (see [XML], 2.10 "White Space Handling").
static int	**START_DOCUMENT** Indicates an event is a start document
static int	**START_ELEMENT** Indicates an event is a start element

Field Detail

START_ELEMENT

`static final int` **`START_ELEMENT`**

Indicates an event is a start element

See Also:
`StartElement`, Constant Field Values

END_ELEMENT

`static final int` **`END_ELEMENT`**

Indicates an event is an end element

See Also:
`EndElement`, Constant Field Values

PROCESSING_INSTRUCTION

`static final int` **`PROCESSING_INSTRUCTION`**

Indicates an event is a processing instruction

See Also:
`ProcessingInstruction`, Constant Field Values

CHARACTERS

`static final int` **`CHARACTERS`**

Indicates an event is characters

See Also:
`Characters`, Constant Field Values

COMMENT

`static final int` **`COMMENT`**

Indicates an event is a comment

See Also:
`Comment`, Constant Field Values

SPACE

`static final int` **`SPACE`**

The characters are white space (see [XML], 2.10 "White Space Handling"). Events are only reported as SPACE if they are ignorable white space. Otherwise they are reported as CHARACTERS.

See Also:
`Characters`, Constant Field Values

START_DOCUMENT

`static final int` **`START_DOCUMENT`**

Indicates an event is a start document

See Also:
`StartDocument`, Constant Field Values

END_DOCUMENT

`static final int` **`END_DOCUMENT`**

Indicates an event is an end document

See Also:
`EndDocument`, Constant Field Values

ENTITY_REFERENCE

`static final int` **`ENTITY_REFERENCE`**

Indicates an event is an entity reference

See Also:
`EntityReference`, Constant Field Values

ATTRIBUTE

`static final int` **`ATTRIBUTE`**

Indicates an event is an attribute

See Also:
`Attribute`, Constant Field Values

DTD

`static final int` **`DTD`**

Indicates an event is a DTD

See Also:
`DTD`, Constant Field Values

CDATA

`static final int` **`CDATA`**

Indicates an event is a CDATA section

See Also:
`Characters`, Constant Field Values

NAMESPACE

`static final int` **`NAMESPACE`**

Indicates the event is a namespace declaration

See Also:
`Namespace`, Constant Field Values

NOTATION_DECLARATION

`static final int` **`NOTATION_DECLARATION`**

Indicates a Notation

See Also:
`NotationDeclaration`, Constant Field Values

ENTITY_DECLARATION

`static final int` **`ENTITY_DECLARATION`**

Indicates a Entity Declaration

See Also:

`NotationDeclaration`, Constant Field Values

---

we can classify these methods into

1 .Element related
2. Attribute related
3. Character related (Text / CDATA / Comment)
4. Namespace related
5. Document related
6. streamReader related

StAX API dont have any special method to call for parsing..... super design .. because Clint only has control of thread

21 - Nov - 08

27 - Dec - 08

javax.xml.strea

Interface XMLStreamReader

46 - methods

6 - document related methods
7 - Element related methods
9 - attribute related methods (getter)
7 - Character (comment / CDATA / text) mtds
6 - namespace related methods (getter)
7 - reader related methods

if Stream / Event reader dont have handy method to check event types , we can only check by comparing current event type with XMLStreamConstants

All

All Known Implementing Classes:

StreamReaderDelegate

Every StartElement will have its own NamespaceContext

ALL Attribute related methods are start with "attribute" word.

```
public interface XMLStreamReader
```

extends XMLStreamConstants

All Characters event related methods will have "Text" word in method

EVENT VALUE

START_ELEMENT = 1
END_ELEMNET = 2

CHARACTERS = 4 (CDATA / text)

COMMENT = 5

START_DOCUMENT = 7
END_DOCUMENT = 8

ATTRIBUTE = 10

NAMESPACE = 13

The XMLStreamReader interface allows forward, read-only access to XML. It is efficient way to read XML data.

The XMLStreamReader is designed to iterate over XML using next() and hasNext such as getEventType(), getNamespaceURI(), getLocalName() and getText();

The next() method causes the reader to read the next parse event. The next() meth of event just read.

The event type can be determined using getEventType().

USE Switch - case to evaluate each type of events.

use one case to each eventype

```
// use to evaluate Attribute event related methods
private void attributeMethods(XMLStreamReader xmlStreamReader) { }
// use to evaluate Element event related methods
private void elementMethods(XMLStreamReader xmlStreamReader) { }
// use to evaluate Character( Comment, CDATA, Text ) event related methods
private void characterMethods(XMLStreamReader xmlStreamReader) { }
// use to evaluate Document event related methods
private void documentMethods(XMLStreamReader xmlStreamReader) { }
// use to evaluate GENERAL  related methods
private void generalMethods(XMLStreamReader xmlStreamReader) { }
// use to evaluate nameSpace event related methods
private void nameSpaceMethods(XMLStreamReader xmlStreamReader) { }
```

~~Parsing e processin query ope~~

~~For XML their asso following current ev~~ nd the

xml.st

getPro ~~are defin~~ nd

The follo throw a java.lang.IllegalStateException. ll

Valid methods for each state

Event Type	Valid Methods
All States	getProperty(), hasNext(), require(), close(), getNamespaceURI(), isStartElement(), isEndElement(), isCharacters(), isWhiteSpace(), getNamespaceContext(), getEventType (),getLocation(), hasText(), hasName()
START_ELEMENT	next(), getName(), getLocalName(), hasName(), getPrefix(), getAttributeXXX(), isAttributeSpecified(), getNamespaceXXX(), getElementText(), nextTag()
ATTRIBUTE	next(), nextTag() getAttributeXXX(), isAttributeSpecified(),
NAMESPACE	next(), nextTag() getNamespaceXXX()
END_ELEMENT	next(), getName(), getLocalName(), hasName(), getPrefix(), getNamespaceXXX(), nextTag()
CHARACTERS	next(), getTextXXX(), nextTag()
CDATA	next(), getTextXXX(), nextTag()
COMMENT	next(), getTextXXX(), nextTag()
SPACE	next(), getTextXXX(), nextTag()
START_DOCUMENT	next(), getEncoding(), getVersion(), isStandalone(), standaloneSet(), getCharacterEncodingScheme(), nextTag()
END_DOCUMENT	close()
PROCESSING_INSTRUCTION	next(), getPITarget(), getPIData(), nextTag()

USE Switch - case to evaluate each type of events.

use one case to each eventype

```
// use to evaluate Attribute event related methods
 private void attributeMethods(XMLStreamReader xmlStreamReader) { }
 // use to evaluate Element event related methods
 private void elementMethods(XMLStreamReader xmlStreamReader) { }
 // use to evaluate Character( Comment, CDATA, Text ) event related methods
 private void characterMethods(XMLStreamReader xmlStreamReader) { }
 // use to evaluate Document event related methods
 private void documentMethods(XMLStreamReader xmlStreamReader) { }
 // use to evaluate GENERAL  related methods
 private void generalMethods(XMLStreamReader xmlStreamReader) { }
 // use to evaluate nameSpace event related methods
 private void nameSpaceMethods(XMLStreamReader xmlStreamReader) { }
```

Field Summary

Fields inherited from interface javax.xml.stream.XMLStreamConstants

ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Method Summary

R void	**close**() ~~Frees any resources associated with this Reader~~.
A int	**getAttributeCount**() ~~Returns the count of attributes on this START_ELEMENT, this method is only valid on a START_ELEMENT or ATTRIBUTE.~~
A java.lang.String	**getAttributeLocalName**(int index) ~~Returns the localName of the attribute at the provided index~~
A QName	**getAttributeName**(int index) ~~Returns the qname of the attribute at the provided index~~
A java.lang.String	**getAttributeNamespace**(int index) ~~Returns the namespace of the attribute at the provided index~~
A java.lang.String	**getAttributePrefix**(int index) ~~Returns the prefix of this attribute at the provided index~~
A java.lang.String	**getAttributeType**(int index) ~~Returns the XML type of the attribute at the provided index~~
A java.lang.String	**getAttributeValue**(int index) ~~Returns the value of the attribute at the index~~
A java.lang.String	**getAttributeValue**(java.lang.String namespaceURI, java.lang.String localName) ~~Returns the normalized attribute value of the attribute~~ ~~If the namespaceURI is null the namespace is not checked for equality~~
D java.lang.String	**getCharacterEncodingScheme**() Returns the character encoding declared on the xml declaration Returns null if none was declared
E java.lang.String	**getElementText**() Reads the content of a text-only element, an exception is thrown if this is not a text-only element.
D java.lang.String	**getEncoding**() ~~Return input encoding i~~
R int	**getEventType**() ~~Returns an integer code that indicates the type of the event the cursor is pointing to~~.
R java.lang.String	**getLocalName**() ~~Returns the (local) name of the current event~~.
R Location	**getLocation**() ~~Return the current location of the processor.~~
E QName	**getName**() ~~Returns a QName for the current START_ELEMENT or END_ELEMENT event~~

ALL of these Attribute related methods are INDEX based only

SUN would have named like "isCharacterEncodingSchemSpecified()

so, Mixed content cannot be accessed

so, dont try to get text of element EXCEPT CHARACTER event. correctaaaaa !!!

NameSpaceContext and QName are different object not from StAX API

NamespaceContext	**getNamespaceContext**() ~~Returns a read only namespace con~~
int	**getNamespaceCount**() ~~Returns the count of namespaces declared on this START_ELEMENT or END_ELEMENT, this method is only valid on a START_ELEMENT, END_ELEMENT or NAMESPACE.~~
java.lang.String	**getNamespacePrefix**(int index) ~~Returns the prefix for the namespace declared at the index.~~
java.lang.String	**getNamespaceURI**() ~~If the current event is a START_ELEMENT or END_ELEMENT this method returns the URI of the prefix or the default namespace.~~
java.lang.String	**getNamespaceURI**(int index) ~~Returns the uri for the namespace declared at the index.~~
java.lang.String	**getNamespaceURI**(java.lang.String prefix) ~~Return the uri for the given prefix.~~
~~java.lang.String~~	~~**getPIData**()~~ ~~Get the data section of a processing instruction~~
~~java.lang.String~~	~~**getPITarget**()~~ ~~Get the target of a processing instruction~~
java.lang.String	**getPrefix**() ~~Returns the prefix of the current event or null if the event does not have a prefix~~
java.lang.Object	**getProperty**(java.lang.String name) ~~Get the value of a feature/property from the underlying implementation~~
java.lang.String	**getText**() Returns the current value of the parse event as a string, this returns the string value of a CHARACTERS event, returns the value of a COMMENT, ~~the replacement value for an ENTITY_REFERENCE~~, the string value of a CDATA section, the string value for a SPACE event, ~~or the String value of the internal subset of the DTD.~~
char[]	**getTextCharacters**() ~~Returns an array which contains the characters from this event.~~
int	**getTextCharacters**(int sourceStart, char[] target, int targetStart, int length) Gets the the text associated with a CHARACTERS, SPACE or CDATA event.
int	**getTextLength**() Returns the length of the sequence of characters for this Text event within the text character array.
int	**getTextStart**() Returns the offset into the text character array where the first character (of this text event) is stored.
java.lang.String	**getVersion**() ~~Get the xml version declared on the xml declaration Returns null if none was declared~~

boolean	**hasName**() returns true if the current event has a name (is a START_ELEMENT or END_ELEMENT) returns false otherwise
boolean	**hasNext**() ~~Returns true if~~ ~~ts.~~
boolean	**hasText**() Return true if the current event has text, false otherwise The following events have text: CHARACTERS,~~DTD~~ ,~~ENTITY_REFERENCE~~, COMMENT, SPACE
boolean	**isAttributeSpecified**(int index) ~~Returns a boolean which indicates if this attribute was created by default~~
boolean	**isCharacters**() Returns true if the cursor
boolean	**isEndElement**() Returns true if the cursor
boolean	**isStandalone**() ~~Get the standalone declara~~
boolean	**isStartElement**() ~~Returns true if the cursor~~
boolean	**isWhiteSpace**() Returns true if the cursor points to a character data event that consists of all whitespace
int	**next**() Get next parsing event - a processor may return all contiguous character data in a single chunk, or it may split it into several chunks.
int	**nextTag**() Skips any white space (isWhiteSpace() returns tr PROCESSING_INSTRUCTION, until a START_ELE d.
void	**require**(int type, java.lang.String namespaceURI, java.lang.String localName) Test if the current event is of the given type and if the namespace and name match the current namespace and name of the current event.
~~boolean~~	~~**standaloneSet**()~~ ~~Checks if standalone was set in the document~~

E

R

C

A

E

D

E

C

R

E

Wow

So, check with hasText() method before TRY to get text of Element (calling getElementText()) for safe from exception

These 4 methods are HANDY methods

1. isStartElement()
2. isEndElement()
3. isCharacters()
4. isWhiteSpace()

XMLEvent object also has 7 handy methods like this

otherwise we have to code like this comparing with constant and current type of cursor

```
if (XMLStreamConstants.START_DOCUMENT ==
xmlStreamReader.getEventType()) {  }
```

it is same as cursor API

we can write our own handy method by WRAPPING this xmlStreamReader by comparing with XMLStreamConstants

we cannot extends. since it is interface. Wrapping is the only way when we want to extend the functionality if a class is out of SCOPE (third party JAXP jar).

Method Detail

getProperty

look XMLInputFactory javadoc for more properties.

```
java.lang.Object getProperty(java.lang.String name)
                 throws java.lang.IllegalArgumentException
```

~~Get the~~

i can use any arbitrary name as property name. (i tested). no exception

~~**Parameters:**~~

`name` ~~- The name of the property, may not be null~~

~~**Returns:**~~

~~The value of the property~~

~~**Throws:**~~

`java.lang.IllegalArgumentException` ~~- if name is null~~

---

next

```
int next()
        throws XMLStreamException
```

Get next parsing event - a processor may return all contiguous character data in a single chunk, or it may split it into several chunks. If the property javax.xml.stream.isCoalescing is set to true element content must be coalesced and only one CHARACTERS event must be returned for contiguous element content or CDATA Sections. ~~By default entity references must be expanded and reported transparently to the application. An exception will be thrown if an entity reference cannot be expanded.~~ If element content is empty (i.e. content is "") then no CHARACTERS event will be reported.

Super

~~Given the following XML:~~
~~<foo><!--description-->content text<![CDATA[<greeting>Hello</greeting>]]>other content</foo>~~
~~The behavior of calling next() when being on foo will be:~~
~~1- the comment (COMMENT)~~
~~2- then the characters section (CHARACTERS)~~
~~3- then the CDATA section (another CHARACTERS)~~
~~4- then the next characters section (another CHARACTERS)~~
~~5- then the END_ELEMENT~~

~~**NOTE:** empty element (such as <tag/>) will be reported with two separate events: START_ELEMENT, END_ELEMENT - This preserves parsing equivalency of empty element to <tag></tag>. This method will throw an IllegalStateException if it is called after hasNext() returns false.~~

Returns:

the integer code corresponding to the current parse event

Throws:

`NoSuchElementException` - if this is called when hasNext() returns false

`XMLStreamException` - if there is an error processing the underlying XML source

See Also:

`XMLEvent`

---

require

this method return nothing.
simply throw exception if not matched any of three parameter. i tested.

```
void require(int type,
             java.lang.String namespaceURI,
             java.lang.String localName)
```

throws XMLStreamException

Test if the current event is of the given type and if the namespace and name match the current namespace and name of the current event. If the namespaceURI is null it is not checked for equality, if the localName is null it is not checked for equality.

Parameters:

`type` - the event type

`namespaceURI` - the uri of the event, may be null

`localName` - the localName of the event, may be null

Throws:

`XMLStreamException` - if the required values are not matched.

so no need this method to use

---

getElementText

```
java.lang.String getElementText()
                                 throws XMLStreamException
```

if it is start element of Nested elements , it will throws expection

Reads the content of a text-only element, an exception is thrown if this is not a text-only element. Regardless of value of javax.xml.stream.isCoalescing this method always returns coalesced content.
Precondition: the current event is START_ELEMENT.
Postcondition: the current event is the corresponding END_ELEMENT.
~~The method does the following (implementations are free to optimized but must do equivalent processing):~~

it can be called only at start element and that element has only TEXT,

So call hasText() method before to call this method

```
if(getE
throw n
"parser must be on START_ELEMENT to read next text", getLocation());
}
int eventType = next();
StringBuffer content = new StringBuffer();
while(eventType != XMLStreamConstants.END_ELEMENT ) {
if(eventType == XMLStreamConstants.CHARACTERS
|| eventType == XMLStreamConstants.CDATA
|| eventType == XMLStreamConstants.SPACE
|| eventType == XMLStreamConstants.ENTITY_REFERENCE) {
buf.append(getText());
} else if(eventType == XMLStreamConstants.PROCESSING_INSTRUCTION
|| eventType == XMLStreamConstants.COMMENT) {
// skipping
} else if(eventType == XMLStreamConstants.END_DOCUMENT) {
throw new XMLStreamException(
"unexpected end of document when reading element text content", this);
} else if(eventType == XMLStreamConstants.START_ELEMENT) {
throw new XMLStreamException(
"element text content may not contain START_ELEMENT", getLocation());
} else {
throw new XMLStreamException(
"Unexpected event type "+eventType, getLocation());
}
```

```
eventType = next();
}
return buf.toString();
```

Throws:

XMLStreamException - if the current event is not a START_ELEMENT or if a non text element is encountered

---

nextTag

```
int nextTag()
        throws XMLStreamException
```

Skips any white space (isWhiteSpace() returns true), COMMENT, or PROCESSING_INSTRUCTION, until a START_ELEMENT or END_ELEMENT is reached. If other than white space characters, COMMENT, PROCESSING_INSTRUCTION, START_ELEMENT, END_ELEMENT are encountered, an exception is thrown. This method should be used when processing element-only content seperated by white space.
Precondition: none

nextTag() method useful if and only if want to move to

1. next sibling's start tag / end tag of its parent **while cursor pointing end tag**
2. next child's start tag **while cursor pointing start tag**

```
<root>(cursor here)[ now cursor move to start of child (<sub>) tag ]

   <sub>text</sub>(cursor here)[ now cursor move to start of sibling <sub> tag ]

   <sub>test</sub>(cursor here)[ now cursor move to end of parent (</root>) tag ]
</root>

if we call the nextTag() method while cursor pointing other than these eventType on
XMLStreamReader it will throw exception only


 ***** so, Try to avoid as much as possible using this method *****
```

```
}
return eventType;
```

Returns:

the event type of the element read (START_ELEMENT or END_ELEMENT)

Throws:

XMLStreamException - if the current event is not white space, PROCESSING_INSTRUCTION, START_ELEMENT or END_ELEMENT

NoSuchElementException - if this is called when hasNext() returns false

---

hasNext

```
boolean hasNext()
                throws XMLStreamException
```

Returns true if there are more parsing events and false if there are no more events. This method will return false if the current state of the XMLStreamReader is END_DOCUMENT

~~**Returns:**~~
~~true if there are more events, false otherwise~~
~~**Throws:**~~
~~XMLStreamException - if there is a fatal error detecting the next state~~

---

close

```
void close()
          throws XMLStreamException
```

Frees any resources associated with this Reader. This method does not close the underlying input source.

Throws:
XMLStreamException - if there are errors freeing associated resources

---

getNamespaceURI

this 'prefix' parametered method can be called at any type event

```
java.lang.String getNamespaceURI(java.lang.String prefix)
```

Return the uri for the given prefix. The uri returned depends on the current state of the processor.

NOTE: The 'xml' prefix is bound as defined in Namespaces in XML specification to "http://www.w3.org/XML/1998/namespace".

NOTE: The 'xmlns' prefix must be resolved to following namespace http://www.w3.org/2000/xmlns/

this method can EVEN be called after XMLStreamReader closed (i tested)

Parameters:
`prefix` - The prefix to lookup, may not be null

```
logger.info("uri of prefix xml ::"+xmlStreamReader.getNamespaceURI("xml"));
  logger.info("uri of prefix xmlns ::"+xmlStreamReader.getNamespaceURI("xmlns"));
```

Throws:
`java.lang.IllegalArgumentException` - if the prefix is null

---

isStartElement

boolean **isStartElement**()

~~Returns tru~~

this is handy method rather than checking like this

```
if(XMLStreamConstants.START_ELEMENT == xmlStreamReader.getEventType()) {
}
```

~~**Returns:**~~

~~true~~

---

isEndElement

boolean **isEndElement**()

~~Returns~~

this is handy method rather than checking like this

```
if(XMLStreamConstants.END_ELEMENT == xmlStreamReader.getEventType()) {
}
```

~~**Returns:**~~

~~tr~~

---

isCharacters

boolean **isCharacters**()

~~Returns tr~~

this is handy method rather than checking like this

```
if(XMLStreamConstants.CHARACTERS == xmlStreamReader.getEventType()) {
}
```

~~**Returns:**~~

~~tru~~

use this code to avoid remove all whitespace while writing out to file / network

```
if(xmlStreamReader.hasText()) {
   if( ! xmlStreamReader.isWhiteSpace()) {
    logger.info(xmlStreamReader.getText());
   }
  }
```

---

isWhiteSpace

boolean **isWhiteSpace**()

Returns true if the cursor points to a character data event that consists of all whitespace

Returns:

true if the cursor points to all whitespace, false otherwise

---

since, ATTRIBUTE is secondary event, access any attribute related method only after got STATE_ELEMENT event

getAttributeValue

java

for safety, check attribute count before to access any value

```
if(xmlStreamReader.getAttributeCount() > 0) {
    logger.info("Attribute value ::"+xmlStreamReader.getAttributeValue(null, "name")); }
```

Returns the normalized attribute value of the attribute with the namespace and localName If the namespaceURI is null

the namespace is not checke

i tried, to use this method at Attribute Event. but, still i got this IllegalStateException. i dont know why ?

But works in START_ELEMENT event.
Since, Attribute is secondary event. So we have to access attribute when cursor event type is START_ELEMENT

Parameters:

`namespaceURI` - th

`localName` - the lo

Returns:

returns the value of the attribute , returns null if not found

Throws:

`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE

---

getAttributeCount

`int` **`getAttributeCount`**`()`

i tried, to use this method at Attribute Event. but, still i got this IllegalStateException. i dont know why ?
but works in StartElement event.

Since it is secondary event, **the same behavior apply to Iterator API Too**

Returns the count of attributes on th ENT or ATTRIBUTE. This count excludes namespace definitions. Attribute indices are zero-based.

Returns:

returns the number of attributes

Throws:

`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ~~ATTRIBUTE~~

---

getAttributeName

`QName` **`getAttributeName`**`(int index)`

access attribute related methods only at START_ELEMENT. it works fine.
Don't try with ATTRIBUTE event, since it is secondary event

Returns the qname of the attribute at the provided index

Paran

this method return non-null if the attribute name is prefixed and namespace

Retur

the QName of the attribute

Throws:

`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE

---

getAttributeNamespace

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

~~Returns the namespace of the attribute at the provided index~~

~~**Parameters:**~~

~~`index` - the position of the attribute~~

Returns:
the namespace URI (can be null) yes
Throws:
`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE

---

getAttributeLocalName

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

~~Returns the localName of the attribute at the provided index~~

~~**Parameters:**~~
~~`index` - the position of the attribute~~
~~**Returns:**~~
~~the localName of the attribute~~
~~**Throws:**~~
~~`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE~~

---

getAttributePrefix

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

it return empty if not attribute is not associated with any prefix

~~Returns the prefix of this attribute at the provided index~~

~~**Parameters:**~~
~~`index` - the position of the attribute~~
~~**Returns:**~~
~~the prefix of the attribute~~
~~**Throws:**~~
~~`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE~~

---

getAttributeType

it return type as "CDATA" .

i dont know on what basis it return this value as type

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

~~Returns the XML type of the attribute at the provided index~~

~~**Parameters:**~~
~~`index` - the position of the attribute~~
~~**Returns:**~~
~~the XML type of the attribute~~
~~**Throws:**~~
~~`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE~~

---

getAttributeValue

since, ATTRIBUTE is secondary event, access any attribute related method only after got STATE_ELEMENT event

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

~~Returns the value of the attribute at the index~~

~~**Parameters:**~~
~~`index` – the position of the attribute~~
~~**Returns:**~~
~~the attribute value~~
~~**Throws:**~~
~~`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE~~

---

isAttributeSpecified

`boolean` **`isAttributeSpecified`**`(int index)`

~~Returns a boolean which indicates if this attribute was created by default~~

~~**Parameters:**~~
~~`index` – the position of the attribute~~
~~**Returns:**~~
~~true if this is a default attribute~~
Throws:
`java.lang.IllegalStateException` - if this is not a START_ELEMENT or ATTRIBUTE

---

getNamespaceCount

it retrun only count of CURRENT element, but not previously visited elements

`int` **`getNamespaceCount`**`()`

Returns the count of namespaces declared on this START_ELEMENT or END_ELEMENT, this method is only valid on a START_ELEMENT, END_ELEMENT or NAMESPACE. On an END_ELEMENT the count is of the namespaces that are about to go out of scope. This is the equivalent of the information reported by SAX callback for an end element event.

Returns:
returns the number of namespace declarations on this specific element
Throws:
`java.lang.IllegalStateException` - if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE

---

getNamespacePrefix

check namespacecount > 0 before to access.

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

~~Returns the prefix for the namespace declared at the index. Returns null if this is the default namespace declaration~~

~~**Parameters:**~~

~~`index` - the position of the namespace declaration~~

~~**Returns:**~~

~~returns the namespace prefix~~

Throws:

`java.lang.IllegalStateException` - if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE

```
if(xmlStreamReader.getNamespaceCount() > 0){

   logger.info(" namesapce  prefix "+xmlStreamReader.getNamespacePrefix(0));
   logger.info(" namesapce  prefix "+xmlStreamReader.getNamespacePrefix(1));
  }
```

---

getNamespaceURI

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

~~Returns the uri for the namespace declared at the index.~~

for this, no need any total count check

~~**Parameters:**~~

~~`index` - the position of the namespace declaration~~

~~**Returns:**~~

~~returns the namespace uri~~

Throws:

`java.lang.IllegalStateException` - if this is not a START_ELEMENT, END_ELEMENT or NAMESPACE

---

getNamespaceContext

only one instance of this object shared the whole life of parsing of SINGLE Document. (i tested for different element it use the same INSTANCE)

`NamespaceContext` **`getNamespaceContext`**`()`

Returns a read only namespace context for the current position. The context is transient and only valid until a call to next () changes the state of the reader.

Returns:

return a namespace context

---

getEventType

`int` **`getEventType`**`()`

~~Returns an integer code that indicates the type of the event the cursor is pointing to~~.

---

getText

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

Returns the current value of the parse event as a string, this returns the string value of a CHARACTERS event, returns the value of a COMMENT, the replacement value for an ENTITY_REFERENCE, the string value of a CDATA section, the string value for a SPACE event, or the String value of the internal subset of the DTD. If an ENTITY_REFERENCE has been resolved, any character data will be reported as CHARACTERS events.

Returns:
the current text or null
Throws:
`java.lang.IllegalStateException` - if this state is not a valid text state.

---

getTextCharacters

it can be called only on TEXT/CDATA/ COMMENT event.

`char[]` **`getTextCharacters`**`()`

Returns an array which contains the characters from this event. This array should be treated as read-only and transient. I. e. the array will contain the text characters until the XMLStreamReader moves on to the next event. Attempts to hold onto the character array beyond that time or modify the contents of the array are breaches of the contract for this interface.

Returns:
the current text or an empty array
Throws:
`java.lang.IllegalStateException` - if this state is not a valid text state.

---

getTextCharacters

it can be called only on TEXT/CDATA/ COMMENT event.

```
int getTextCharacters(int sourceStart,
                      char[] target,
                      int targetStart,
                      int length)
                      throws XMLStreamException
```

Gets the the text associated with a CHARACTERS, SPACE or CDATA event. Text starting a "sourceStart" is copied into "target" starting at "targetStart". Up to "length" characters are copied. The number of characters actually copied is returned. The "sourceStart" argument must be greater or equal to 0 and less than or equal to the number of characters associated with the event. Usually, one requests text starting at a "sourceStart" of 0. If the number of characters actually copied is less than the "length", then there is no more text. Otherwise, subsequent calls need to be made until all text has been retrieved. For example: `int length = 1024; char[] myBuffer = new char[ length ]; for ( int sourceStart = 0 ; ; sourceStart += length ) { int nCopied = stream.`

`getTextCharacters( sourceStart, myBuffer, 0, length ); if (nCopied < length) break; }` XMLStreamException may be thrown if there are any XML errors in the underlying source. The "targetStart" argument must be greater than or equal to 0 and less than the length of "target", Length must be greater than 0 and "targetStart + length" must be less than or equal to length of "target".

Parameters:

`sourceStart` - the index of the first character in the source array to copy
`target` - the destination array
`targetStart` - the start offset in the target array
`length` - the number of characters to copy

Returns:

the number of characters actually copied

Throws:

`XMLStreamException` - if the underlying XML source is not well-formed
`java.lang.IndexOutOfBoundsException` - if targetStart < 0 or > than the length of target
`java.lang.IndexOutOfBoundsException` - if length < 0 or targetStart + length > length of target
`java.lang.UnsupportedOperationException` - if this method is not supported
`java.lang.NullPointerException` - is if target is null

---

getTextStart

it can be called only on TEXT/CDATA/ COMMENT event.

`int getTextStart()`

Returns the offset into the text character array where the first character (of this text event) is stored.

Throws:

`java.lang.IllegalStateException` - if this state is not a valid text state.

---

getTextLength

it can be called only on TEXT/CDATA/ COMMENT event.

`int getTextLength()`

~~Returns the length of the sequence of characters for this Text event within the text character array.~~

Throws:

`java.lang.IllegalStateException` - if this state is not a valid text state.

---

getEncoding

`java.lang.String getEncoding()`

~~Return input encoding if known or null if unknown.~~

Returns:
~~the encoding of this instance or null~~

---

hasText

check with this method before try to get text content of a ELEMENT

`boolean` **`hasText()`**

Return true if the current event has text, false otherwise The following events have text: CHARACTERS,~~DTD~~, ~~ENTITY_REFERENCE~~, COMMENT, SPACE

---

getLocation

they have overridden toString method of Location. it is good

`Location` **`getLocation()`**

Return the current location of the processor. If the Location is unknown the processor should return an implementation of Location that returns -1 for the location and null for the publicId and systemId. The location information is only valid until next() is called.

---

getName

`QName` **`getName()`**

Returns a QName for the current START_ELEMENT or END_ELEMENT event

Returns:
the QName for the current START_ELEMENT or END_ELEMENT event
Throws:
`java.lang.IllegalStateException` - if this is not a START_ELEMENT or END_ELEMENT

---

getLocalName

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

Returns the (local) name of the current event. For START_ELEMENT or END_ELEMENT returns the (local) name of the current element. For ENTITY_REFERENCE it returns entity name. The current event must be START_ELEMENT or END_ELEMENT, or ENTITY_REFERENCE

Returns:
the localName
Throws:
`java.lang.IllegalStateException` - if this not a START_ELEMENT, END_ELEMENT or

ENTITY_REFERENCE

---

use this method before try to get name of Element. mostly we use this to print in screen

hasName

`boolean hasName()`

returns true if the current event has a name (is a START_ELEMENT or END_ELEMENT) returns false otherwise

---

if there is no prefix in element name even it has namespace declration it will return null only

getNamespaceURI

`java.lang.String getNamespaceURI()`

If the current event is a START_ELEMENT or END_ELEMENT this method returns the URI of the prefix or the default namespace. Returns null if the event does not have a prefix.

Returns:
the URI bound to this elements prefix, the default namespace, or null

---

getPrefix

`java.lang.String getPrefix()`

start / end element

Returns the prefix of the current event or null if the event does not have a prefix

Returns:
the prefix or null

---

getVersion

`java.lang.String getVersion()`

~~Get the xml version declared on the xml declaration Returns null if none was declared~~

Returns:
the XML version or null

---

this standalone is related to DTD

~~isStandalone~~

~~`boolean isStandalone()`~~

~~Get the standalone declaration from the xml declaration~~

~~**Returns:**~~
~~true if this is standalone, or false otherwise~~

~~standaloneSet~~

~~`boolean` **`standaloneSet()`**~~

~~Checks if~~ standalone was set in the document

Returns:
true if standalone was set in the document, or false otherwise

---

getCharacterEncodingScheme

why this method.
already encoding method is there

simply it return null.

it is for, whether encoding is specified on XML document. method name has to be changed

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

Returns the character encoding declared on the xml declaration

Returns:
the encoding declared in the document or null

~~getPITarget~~

~~`java.lang.String` **`getPITarget()`**~~

DTD

~~Get the target of a processing instruction~~

~~**Returns:**~~
~~the target or null~~

~~getPIData~~

~~`java.lang.String` **`getPIData()`**~~

DTD

~~Get the data section of a processing instruction~~

~~**Returns:**~~
~~the data or null~~

21 - Nov - 08

28 - Dec - 08

we can classify these methods into

1 .Element related
2. Attribute related
3. Character related (Text / CDATA / Comment)
4. Namespace related
5. Document related
6. streamWriter related

28 methods

7 - methods for ELEMENT
3 - methods for ATTRIBUTE
4 - methods for CHARACTER
7 - methods for NAMESPACE
4 - methods for DOCUMENT
3 - Writer related methods

javax.xml.stream

Interface XMLStreamW

`public` Every StartElement will have its own NamespaceContext

toString() has not been overrided. so cannot use instance with SOP or logger

The XMLStreamWriter interface specifies how to write XML. The XMLStreamWriter does not perform well formedness checking on its input. However the writeCharacters method is required to escape & , < and > For attribute values the writeAttribute method will escape the above characters plus " to ensure that all character content and attribute values are well formed. Each NAMESPACE and ATTRIBUTE must be individually written.

XML Namespaces, `javax.xml.stream.isRepairingNamespaces` and write method behaviour				
Method	**`isRepairingNamespaces` == true**		**`isRepairingNamespaces` == false**	
	namespaceURI bound	**namespaceURI unbound**	**namespaceURI bound**	**namespaceURI unbound**
`writeAttribute (namespaceURI, localName, value)`	prefix:localName="value" [1]	xmlns:{generated} ="namespaceURI" {generated}: localName="value"	prefix: localName="value" [1]	`XMLStreamException`
`writeAttribute (prefix, namespaceURI, localName, value)`	bound to same prefix: prefix:localName="value" [1] bound to different prefix: xmlns:{generated} ="namespaceURI" {generated}: localName="value"	xmlns:prefix="namespaceURI" prefix:localName="value" [3]	bound to same prefix: prefix: localName="value" [1][2] bound to different prefix: `XMLStreamException` [2]	xmlns: prefix="namespaceURI" prefix: localName="value" [2][5]
`writeStartElement (namespaceURI, localName)` **`writeEmptyElement (namespaceURI, localName)`**	<prefix:localName> [1]	<{generated}:localName xmlns: {generated} ="namespaceURI">	<prefix:localName> [1]	`XMLStreamException`
`writeStartElement (prefix, localName, namespaceURI)` **`writeEmptyElement (prefix, localName, namespaceURI)`**	bound to same prefix: <prefix:localName> [1] bound to different prefix: <{generated}:localName xmlns: {generated} ="namespaceURI">	<prefix:localName xmlns: prefix="namespaceURI"> [4]	bound to same prefix: <prefix:localName> [1] bound to different prefix: `XMLStreamException`	<prefix:localName>

Notes:

- [1] if namespaceURI == default Namespace URI, then no prefix is written
- [2] if prefix == "" || null && namespaceURI == "", then no prefix or Namespace declaration is generated or written
- [3] if prefix == "" || null, then a prefix is randomly generated
- [4] if prefix == "" || null, then it is treated as the default Namespace and no prefix is generated or written, an xmlns declaration is generated and written if the namespaceURI is unbound
- [5] if prefix == "" || null, then it is treated as an invalid attempt to define the default Namespace and an XMLStreamException is thrown

Since:
1.6
Version:
1.0
Author:

See Also:
`XMLOutputFactory`, `XMLStreamReader`

Method Summary

W void	**close**() ~~Close this writer and free any resources associated with the writer.~~
W void	**flush**() ~~Write any cached data to the underlying output mechanism.~~
N NamespaceContext	**getNamespaceContext**() ~~Returns the current namespace context.~~
N java.lang.String	**getPrefix**(java.lang.String uri) ~~Gets the prefix the uri is bound to~~
W java.lang.Object	**getProperty**(java.lang.String name) ~~Get the value of a feature/property from the underlying implementation~~
N void	**setDefaultNamespace**(java.lang.String uri) Binds a URI to the default namespace This URI is bound in the scope of the current START_ELEMENT / END_ELEMENT pair.
N void	**setNamespaceContext**(NamespaceContext context) Sets the current namespace context for prefix and uri bindings.
N void	**setPrefix**(java.lang.String prefix, java.lang.String uri) Sets the prefix the uri is bound to.
A void	**writeAttribute**(java.lang.String localName, java.lang.String value) Writes an attribute to the output stream without a prefix.
A void	**writeAttribute**(java.lang.String namespaceURI, java.lang.String localName, java.lang.String value) Writes an attribute to the output stream
A void	**writeAttribute**(java.lang.String prefix, java.lang.String namespaceURI, java.lang.String localName, java.lang.String value) Writes an attribute to the output stream
C void	**writeCData**(java.lang.String data) Writes a CData section
C void	**writeCharacters**(char[] text, int start, int len) Write text to the output
C void	**writeCharacters**(java.lang.String text) Write text to the output
C void	**writeComment**(java.lang.String data) Writes an xml comment with the data enclosed
N void	**writeDefaultNamespace**(java.lang.String namespaceURI) Writes the default namespace to the stream

the prefix for this one will be EMPTY

void	~~**writeDTD**(java.lang.String dtd)~~ ~~Write a DTD section.~~
void	**writeEmptyElement**(java.lang.String localName) Writes an empty element tag to the output
void	**writeEmptyElement**(java.lang.String namespaceURI, java.lang.String localName) Writes an empty element tag to the output
void	**writeEmptyElement**(java.lang.String prefix, java.lang.String localName, java.lang.String namespaceURI) ~~Writes an empty element tag to the output~~
void	**writeEndDocument**() Closes any start tags and writes corresponding end tags.
void	**writeEndElement**() Writes an end tag to the output relying on the internal state of the writer to determine the prefix and local name of the event.
void	~~**writeEntityRef**(java.lang.String name)~~ ~~Writes an entity reference~~
void	**writeNamespace**(java.lang.String prefix, java.lang.String namespaceURI) Writes a namespace to the output stream If the prefix argument to this method is the empty string, "xmlns", or null this method will delegate to writeDefaultNamespace
void	~~**writeProcessingInstruction**(java.lang.String target)~~ ~~Writes a processing instruction~~
void	~~**writeProcessingInstruction**(java.lang.String target, java.lang.String data)~~ ~~Writes a processing instruction~~
void	**writeStartDocument**() Write the XML Declaration.
void	**writeStartDocument**(java.lang.String version) Write the XML Declaration.
void	**writeStartDocument**(java.lang.String encoding, j Write the XML Declaration.
void	**writeStartElement**(java.lang.String localName) Writes a start tag to the output.
void	**writeStartElement**(java.lang.String namespaceURI, java.lang.String localName) Writes a start tag to the output
void	**writeStartElement**(java.lang.String prefix, java.lang.String localName, java.lang.String namespaceURI) Writes a start tag to the output

there is no way to add standalone ??????

leave it param. it is DTD related...

Method Detail

writeStartElement

```
void writeStartElement(java.lang.String localName)
                       throws XMLStreamException
```

Writes a start tag to the output. All writeStartElement methods open a new scope in the internal namespace context. Writing the corresponding EndElement causes the scope to be closed.

Parameters:

`localName` - local name of the tag, may not be null

Throws:
`XMLStreamException`

---

writeStartElement

to use this method, on factory isRepairingNamespace property MUST be set to TRUE. otherwise exception will be thrown

```
void writeStartElement(java.lang.String namespaceURI,
                       java.lang.String localName)
                       throws XMLStreamException
```

Writes a start tag to the output

if we set true, the prefix automatically generated, if the prefix is not found

Parameters:
`namespaceURI` - the namespaceURI of the prefix to use, may not be null
`localName` - local name of the tag, may not be null

Throws:
`XMLStreamException` - if the namespace URI has not been bound to a prefix and javax.xml.stream.isRepairingNamespaces has not been set to true

---

writeStartElement

```
void writeStartElement(java.lang.String prefix,
                       java.lang.String localName,
                       java.lang.String namespaceURI)
                       throws XMLStreamException
```

we can write xml file without WELL-FORMED.

Ex: two tag can be in document without having their root tag

By default, StreamWriter NEVER check well-formness of xml

Writes a start tag to the output

Parameters:
`localName` - local name of the tag, may not be null
`prefix` - the prefix of the tag, may not be null
`namespaceURI` - the uri to bind the prefix to, may not be null

Throws:
`XMLStreamException`

calling writeEndElement() on StreamWriter for every startElement() is not mandatory. (i tested)

---

writeEmptyElement

```
void writeEmptyElement(java.lang.String namespaceURI,
                       java.lang.String localName)
                       throws XMLStreamException
```

Writes an empty element tag to the output

Parameters:
`namespaceURI` - the uri to bind the tag to, may not be null
`localName` - local name of the tag, may not be null

Throws:
`XMLStreamException` - if the namespace URI has not been bound to a prefix and javax.xml.stream.isRepairingNamespaces has not been set to true

---

writeEmptyElement

```
void writeEmptyElement(java.lang.String prefix,
                       java.lang.String localName,
                       java.lang.String namespaceURI)
                       throws XMLStreamException
```

~~Writes an empty element tag to the output~~

~~Parameters:~~

`prefix` ~~- the prefix of the tag, may not be null~~

`localName` ~~- local name of the tag, may not be null~~

`namespaceURI` ~~- the uri to bind the tag to, may not be null~~

Throws:

`XMLStreamException`

---

writeEmptyElement

```
void writeEmptyElement(java.lang.String localName)
                       throws XMLStreamException
```

~~Writes an empty element tag to the output~~

~~Parameters:~~

`localName` ~~- local name of the tag, may not be null~~

Throws:

`XMLStreamException`

---

writeEndElement

```
void writeEndElement()
                     throws XMLStreamException
```

Writes an end tag to the output relying on the intern~~al state~~ of the writer to determine the prefix and local name of the event.

Throws:

`XMLStreamException`

---

writeEndDocument

```
void writeEndDocument()
                      throws XMLStreamException
```

Closes any start tags and writes corresponding end tags.

Throws:

`XMLStreamException`

it is not mandatory to call this method, since StAX not check well formness of xml
i tested
StartDocument is also not mandatory

---

close

```
void close()
           throws XMLStreamException
```

Close this writer and free any resources associated with the writer. This must not close the underlying output stream.

Throws:
`XMLStreamException`

if we want to add any think can add after this xml file . yes i tested

---

flush

```
void flush()
           throws XMLStreamException
```

if any one of close() or flush() method does not called, xml will not send to underlying output stream

~~Write any cached data to the underlying output mechanism~~.

Throws:
`XMLStreamException`

---

writeAttribute

```
void writeAttribute(java.lang.String localName,
                    java.lang.String value)
                    throws XMLStreamException
```

to add attribute to empty tag, just call next "writeAttribute" method. since there is not special endElement call for empty tag

Writes an attribute to the output stream without a prefix.

Parameters:
`localName` - the local name of the attribute
`value` - the value of the attribute

Throws:
`java.lang.IllegalStateException` - if the current state does not allow Attribute writing
`XMLStreamException`

these method, dont throw exception only if Most Recent event is StartElement / emptyElement

---

writeAttribute

```
void writeAttribute(java.lang.String prefix,
                    java.lang.String namespaceURI,
                    java.lang.String localName,
                    java.lang.String value)
                    throws XMLStreamException
```

Writes an attribute to the output stream

Parameters:
`prefix` - the prefix for this attribute
`namespaceURI` - the uri of the prefix for this attribute
`localName` - the local name of the attribute
`value` - the value of the attribute

Throws:

`java.lang.IllegalStateException` - if the current state does not allow Attribute writing
`XMLStreamException` - if the namespace URI has not been bound to a prefix and javax.xml.stream.isRepairingNamespaces has not been set to true

---

writeAttribute

```
void writeAttribute(java.lang.String namespaceURI,
                    java.lang.String localName,
                    java.lang.String value)
                    throws XMLStreamException
```

Writes an attribute to the output stream

Parameters:
`namespaceURI` - the uri of the prefix for this attribute
`localName` - the local name of the attribute
`value` - the value of the attribute

Throws:
`java.lang.IllegalStateException` - if the current state does not allow Attribute writing
`XMLStreamException` - if the namespace URI has not been bound to a prefix and javax.xml.stream.isRepairingNamespaces has not been set to true

---

writeNamespace

must be called after START Element only

```
void writeNamespace(java.lang.String prefix,
                    java.lang.String namespaceURI)
                    throws XMLStreamException
```

Writes a namespace to the output stream If the prefix argument to this method is the empty string, "xmlns", or null this method will delegate to writeDefaultNamespace

Parameters:
`prefix` - the prefix to bind this namespace to
`namespaceURI` - the uri to bind the prefix to

Throws:
`java.lang.IllegalStateException` - if the current state does not allow Namespace writing
`XMLStreamException`

---

writeDefaultNamespace

it WONT assign any prefix to this namespace uri.

even TWO namespace can have default namespace (i .e) white space (i tested) in same tag

```
void writeDefaultNamespace(java.lang.String namespaceURI)
                           throws XMLStreamException
```

Writes the default namespace to the stream

Parameters:
`namespaceURI` - the uri to bind the default namespace to

Throws:
`java.lang.IllegalStateException` - if the current state does not allow Namespace writing
`XMLStreamException`

---

writeComment

```
void writeComment(java.lang.String data)
                  throws XMLStreamException
```

~~Writes an xml comment with the data enclosed~~

~~Parameters:~~

`data` ~~- the data contained in the comment, may be null~~

Throws:

`XMLStreamException`

~~writeProcessingInstruction~~

```
void writeProcessingInstruction(java.lang.String target)
                                throws XMLStreamException
```

~~Writes a processing instruction~~

~~Parameters:~~

`target` ~~- the target of the processing instruction, may not be null~~

~~Throws:~~

`XMLStreamException`

~~writeProcessingInstruction~~

```
void writeProcessingInstruction(java.lang.String target,
                                java.lang.String data)
                                throws XMLStreamException
```

~~Writes a processing instruction~~

~~Parameters:~~

`target` ~~- the target of the processing instruction, may not be null~~

`data` ~~- the data contained in the processing instruction, may not be null~~

~~Throws:~~

`XMLStreamException`

---

writeCData

```
void writeCData(java.lang.String data)
                throws XMLStreamException
```

~~Writes a CData section~~

~~Parameters:~~

`data` ~~- the data contained in the CData Section, may not be null~~

Throws:

`XMLStreamException`

~~writeDTD~~

```
void writeDTD(java.lang.String dtd)
              throws XMLStreamException
```

~~Write a DTD section. This string represents the entire doctypedecl production from the XML 1.0 specification.~~

~~Parameters:~~
`dtd` ~~- the DTD to be written~~

~~Throws:~~
`XMLStreamException`

~~writeEntityRef~~

```
void writeEntityRef(java.lang.String name)
                    throws XMLStreamException
```

~~Writes an entity reference~~

~~Parameters:~~
`name` ~~- the name of the entity~~

~~Throws:~~
`XMLStreamException`

---

writeStartDocument

```
void writeStartDocument()
                        throws XMLStreamException
```

it is not mandatory to call this method, since StAX not check well formness of xml
i tested

Write the XML Declaration. Defaults the XML version to 1.0, and the encoding to utf-8

Throws:
`XMLStreamException`

---

writeStartDocument

```
void writeStartDocument(java.lang.String version)
                        throws XMLStreamException
```

i can give any arbitrary version like 5.0 (i tested)

Write the XML Declaration. Defaults the XML version to 1.0

Parameters:
`version` - version of the xml document

Throws:
`XMLStreamException`

---

writeStartDocument

i could not run.
i am getting excepiton.

fix atleast at 5th round (27 - Dec - 08)

```
void writeStartDocument(java.lang.String encoding,
                        java.lang.String version)
                        throws XMLStreamException
```

Write the XML Declaration. Note that the encoding parameter does not set the actual encoding of the underlying output. That must be set when the instance of the XMLStreamWriter is created using the XMLOutputFactory

Parameters:
`encoding` - encoding of the xml declaration
`version` - version of the xml document

Throws:
`XMLStreamException` - If given encoding does not match encoding of the underlying stream

---

writeCharacters

```
void writeCharacters(java.lang.String text)
                     throws XMLStreamException
```

Write text to the output

Parameters:
`text` - the value to write

Throws:
`XMLStreamException`

---

writeCharacters

```
void writeCharacters(char[] text,
                     int start,
                     int len)
                     throws XMLStreamException
```

Write text to the output

Parameters:
`text` - the value to write
`start` - the starting position in the array
`len` - the number of characters to write

Throws:
`XMLStreamException`

---

getPrefix

```
java.lang.String getPrefix(java.lang.String uri)
                           throws XMLStreamException
```

Gets the prefix the uri is bound to

Returns:
the prefix or null

Throws:
XMLStreamException

---

setPrefix

```
void setPrefix(java.lang.String prefix,
               java.lang.String uri)
               throws XMLStreamException
```

Sets the prefix the uri is bound to. This prefix is bound in the scope of the current START_ELEMENT / END_ELEMENT pair. If this method is called before a START_ELEMENT has been written the prefix is bound in the root scope.

Parameters:
`prefix` - the prefix to bind to the uri, may not be null
`uri` - the uri to bind to the prefix, may be null

Throws:
XMLStreamException

---

setDefaultNamespace

what is difference bwn writeDefaultNamespace() and setDefaultNamespace() ???

```
void setDefaultNamespace(java.lang.String uri)
                         throws XMLStreamException
```

Binds a URI to the default namespace This URI is bound in the scope of the current START_ELEMENT / END_ELEMENT pair. If this method is called before a START_ELEMENT has been written the uri is bound in the root scope.

Parameters:
`uri` - the uri to bind to the default namespace, may be null

Throws:
XMLStreamException

---

setNamespaceContext

i dont think, i will have work with this namespace context object ..

```
void setNamespaceContext(NamespaceContext context)
                         throws XMLStreamException
```

Sets the current namespace context for prefix and uri bindings. This context becomes the root namespace context for writing and will replace the current root namespace context. Subsequent calls to setPrefix and setDefaultNamespace will bind namespaces using the context passed to the method as the root context for resolving namespaces. This method may only be called once at the start of the document. It does not cause the namespaces to be declared. If a namespace URI to prefix mapping is found in the namespace context it is treated as declared and the prefix may be used by the StreamWriter.

Parameters:
`context` - the namespace context to use for this writer, may not be null

Throws:
XMLStreamException

---

getNamespaceContext

`NamespaceContext` **`getNamespaceContext()`**

Returns the current namespace context.

Returns:
the current NamespaceContext

---

getProperty

```
java.lang.Object getProperty(java.lang.String name)
                             throws java.lang.IllegalArgumentException
```

Get the value of a feature/property from the underlying implementation

Parameters:
`name` - The name of the property, may not be null
Returns:
The value of the property
Throws:
`java.lang.IllegalArgumentException` - if the property is not supported
`java.lang.NullPointerException` - if the name is null

---

21 - Nov - 08

27 - Dec - 08

1 method

javax.xml.stream

Interface StreamFilter

```
public interface StreamFilter
```

This interface declares a simple filter interface that one can create to filter XMLStreamReaders

so, filter not for writers

Since:
1.6

Version:
1.0

Author:
Copyright (c) 2003 by BEA Systems. All Rights Reserved.

Method Summary	
boolean	**accept**(XMLStreamReader reader) Tests whether the current state is part of this stream.

This is super filter when compare with LS / node Filter.

yes. by this single method we can filter any TYPE of node, not just single type of node by whatToShow

Method Detail

accept

```
boolean accept(XMLStreamReader reader)
```

Tests whether the current state is part of this stream. This method will return true if this filter accepts this event and false otherwise. The method should not change the state of the reader when accepting a state.