org.xml.sax.helpers

Class DefaultHandler

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

All Implemented Interfaces:
ContentHandler, DTDHandler, EntityResolver, ErrorHandler

public class **DefaultHandler**
extends Object
implements EntityResolver, DTDHandler, ContentHandler, ErrorHandler

Default base class for SAX2 event handlers.

*This module, both source code and documentation, is in the Public Domain, and comes with **NO WARRANTY**.*

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

- EntityResolver
- DTDHandler
- ContentHandler
- ErrorHandler

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

This class replaces the deprecated SAX1 HandlerBase class.

Since:
SAX 2.0
See Also:
EntityResolver, DTDHandler, ContentHandler, ErrorHandler

Constructor Summary

DefaultHandler()

Method Summary

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

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

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

Constructor Detail

DefaultHandler

```
public DefaultHandler()
```

Method Detail

resolveEntity

```
public InputSource resolveEntity(String publicId,
                                 String systemId)
                          throws SAXException
```

Resolve an external entity.

Always return null, so that the parser will use the system identifier provided in the XML document. This method implements the SAX default behaviour: application writers can override it in a subclass to do special translations such as catalog lookups or URI redirection.

Specified by:
`resolveEntity` in interface `EntityResolver`

Parameters:
`publicId` - The public identifer, or null if none is available.
`systemId` - The system identifier provided in the XML document.

Returns:
The new input source, or null to require the default behaviour.

Throws:
`IOException` - If there is an error setting up the new input source.
`SAXException` - Any SAX exception, possibly wrapping another exception.

See Also:
`EntityResolver.resolveEntity(java.lang.String, java.lang.String)`

---

notationDecl

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

Receive notification of a notation declaration.

By default, do nothing. Application writers may override this method in a subclass if they wish to keep track of the notations declared in a document.

Specified by:
notationDecl in interface DTDHandler

Parameters:
name - The notation name.
publicId - The notation public identifier, or null if not available.
systemId - The notation system identifier.

Throws:
SAXException - Any SAX exception, possibly wrapping another exception.

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

---

unparsedEntityDecl

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

Receive notification of an unparsed entity declaration.

By default, do nothing. Application writers may override this method in a subclass to keep track of the unparsed entities declared in a document.

Specified by:
unparsedEntityDecl in interface DTDHandler

Parameters:
name - The entity name.
publicId - The entity public identifier, or null if not available.
systemId - The entity system identifier.
notationName - The name of the associated notation.

Throws:
SAXException - Any SAX exception, possibly wrapping another exception.

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

---

setDocumentLocator

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

Receive a Locator object for document events.

By default, do nothing. Application writers may override this method in a subclass if they wish to store the locator for use with other document events.

Specified by:
setDocumentLocator in interface ContentHandler

Parameters:
locator - A locator for all SAX document events.

See Also:
ContentHandler.setDocumentLocator(org.xml.sax.Locator), Locator

---

startDocument

```
public void             ()
```

```
startDocument
        throws SAXException
```

Receive notification of the beginning of the document.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the beginning of a document (such as allocating the root node of a tree or creating an output file).

Specified by:
`startDocument` in interface `ContentHandler`

Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.startDocument()`

---

endDocument

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

Receive notification of the end of the document.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the end of a document (such as finalising a tree or closing an output file).

Specified by:
`endDocument` in interface `ContentHandler`

Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.endDocument()`

---

startPrefixMapping

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

Receive notification of the start of a Namespace mapping.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the start of each Namespace prefix scope (such as storing the prefix mapping).

Specified by:
`startPrefixMapping` in interface `ContentHandler`

Parameters:
`prefix` - The Namespace prefix being declared.
`uri` - The Namespace URI mapped to the prefix.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.startPrefixMapping(java.lang.String, java.lang.String)`

---

endPrefixMapping

```
public void endPrefixMapping(String prefix)
                      throws SAXException
```

Receive notification of the end of a Namespace mapping.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the end of each prefix mapping.

Specified by:
`endPrefixMapping` in interface `ContentHandler`

Parameters:

`prefix` - The Namespace prefix being declared.

Throws:

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

See Also:

`ContentHandler.endPrefixMapping(java.lang.String)`

---

startElement

```
public void startElement(String uri,
                         String localName,
                         String qName,
                         Attributes attributes)
                  throws SAXException
```

Receive notification of the start of an element.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the start of each element (such as allocating a new tree node or writing output to a file).

Specified by:

`startElement` in interface `ContentHandler`

Parameters:

`attributes` - The specified or defaulted attributes.
`localName` - The local name (without prefix), or the empty string if Namespace processing is not being performed.
`qName` - The qualified name (with prefix), or the empty string if qualified names are not available.

Throws:

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

See Also:

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

---

endElement

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

Receive notification of the end of an element.

By default, do nothing. Application writers may override this method in a subclass to take specific actions at the end of each element (such as finalising a tree node or writing output to a file).

Specified by:

`endElement` in interface `ContentHandler`

Parameters:

`localName` - The local name (without prefix), or the empty string if Namespace processing is not being performed.
`qName` - The qualified XML 1.0 name (with prefix), or the empty string if qualified names are not available.

Throws:

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

See Also:

`ContentHandler.endElement(java.lang.String, java.lang.String, java.lang.String)`

---

characters

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

Receive notification of character data inside an element.

By default, do nothing. Application writers may override this method to take specific actions for each chunk of character data (such as adding the data to a node or buffer, or printing it to a file).

Specified by:

`characters` in interface `ContentHandler`

Parameters:
`ch` - The characters.
`start` - The start position in the character array.
`length` - The number of characters to use from the character array.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.characters(char[], int, int)`

---

ignorableWhitespace

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

Receive notification of ignorable whitespace in element content.

By default, do nothing. Application writers may override this method to take specific actions for each chunk of ignorable whitespace (such as adding data to a node or buffer, or printing it to a file).

Specified by:
`ignorableWhitespace` in interface `ContentHandler`

Parameters:
`ch` - The whitespace characters.
`start` - The start position in the character array.
`length` - The number of characters to use from the character array.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.ignorableWhitespace(char[], int, int)`

---

processingInstruction

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

Receive notification of a processing instruction.

By default, do nothing. Application writers may override this method in a subclass to take specific actions for each processing instruction, such as setting status variables or invoking other methods.

Specified by:
`processingInstruction` in interface `ContentHandler`

Parameters:
`target` - The processing instruction target.
`data` - The processing instruction data, or null if none is supplied.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.processingInstruction(java.lang.String, java.lang.String)`

---

skippedEntity

```
public void skippedEntity(String name)
                   throws SAXException
```

Receive notification of a skipped entity.

By default, do nothing. Application writers may override this method in a subclass to take specific actions for each processing instruction, such as setting status variables or invoking other methods.

Specified by:
`skippedEntity` in interface `ContentHandler`

Parameters:
`name` - The name of the skipped entity.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ContentHandler.processingInstruction(java.lang.String, java.lang.String)`

---

warning

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

Receive notification of a parser warning.

The default implementation does nothing. Application writers may override this method in a subclass to take specific actions for each warning, such as inserting the message in a log file or printing it to the console.

Specified by:
`warning` in interface `ErrorHandler`

Parameters:
`e` - The warning information encoded as an exception.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ErrorHandler.warning(org.xml.sax.SAXParseException)`, `SAXParseException`

---

error

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

Receive notification of a recoverable parser error.

The default implementation does nothing. Application writers may override this method in a subclass to take specific actions for each error, such as inserting the message in a log file or printing it to the console.

Specified by:
`error` in interface `ErrorHandler`

Parameters:
`e` - The warning information encoded as an exception.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ErrorHandler.warning(org.xml.sax.SAXParseException)`, `SAXParseException`

---

fatalError

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

Report a fatal XML parsing error.

The default implementation throws a SAXParseException. Application writers may override this method in a subclass if they need to take specific actions for each fatal error (such as collecting all of the errors into a single report): in any case, the application must stop all regular processing when this method is invoked, since the document is no longer reliable, and the parser may no longer report parsing events.

Specified by:
`fatalError` in interface `ErrorHandler`

Parameters:
`e` - The error information encoded as an exception.
Throws:
`SAXException` - Any SAX exception, possibly wrapping another exception.
See Also:
`ErrorHandler.fatalError(org.xml.sax.SAXParseException)`, `SAXParseException`

Submit a bug or feature

For further API reference and developer documentation, see Java 2 SDK SE Developer Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.