| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/XMLEncoder.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/beans/XMLDecoder.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/beans/XMLEncoder.html)    [**NO FRAMES**](http://docs.google.com/XMLEncoder.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

## **java.beans**

Class XMLEncoder

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.beans.Encoder](http://docs.google.com/java/beans/Encoder.html)  
 **java.beans.XMLEncoder**

public class **XMLEncoder**extends [Encoder](http://docs.google.com/java/beans/Encoder.html)

The XMLEncoder class is a complementary alternative to the ObjectOutputStream and can used to generate a textual representation of a *JavaBean* in the same way that the ObjectOutputStream can be used to create binary representation of Serializable objects. For example, the following fragment can be used to create a textual representation the supplied *JavaBean* and all its properties:

XMLEncoder e = new XMLEncoder(  
 new BufferedOutputStream(  
 new FileOutputStream("Test.xml")));  
 e.writeObject(new JButton("Hello, world"));  
 e.close();

Despite the similarity of their APIs, the XMLEncoder class is exclusively designed for the purpose of archiving graphs of *JavaBean*s as textual representations of their public properties. Like Java source files, documents written this way have a natural immunity to changes in the implementations of the classes involved. The ObjectOutputStream continues to be recommended for interprocess communication and general purpose serialization.

The XMLEncoder class provides a default denotation for *JavaBean*s in which they are represented as XML documents complying with version 1.0 of the XML specification and the UTF-8 character encoding of the Unicode/ISO 10646 character set. The XML documents produced by the XMLEncoder class are:

* *Portable and version resilient*: they have no dependencies on the private implementation of any class and so, like Java source files, they may be exchanged between environments which may have different versions of some of the classes and between VMs from different vendors.
* *Structurally compact*: The XMLEncoder class uses a *redundancy elimination* algorithm internally so that the default values of a Bean's properties are not written to the stream.
* *Fault tolerant*: Non-structural errors in the file, caused either by damage to the file or by API changes made to classes in an archive remain localized so that a reader can report the error and continue to load the parts of the document which were not affected by the error.

Below is an example of an XML archive containing some user interface components from the *swing* toolkit:

<?xml version="1.0" encoding="UTF-8"?>  
 <java version="1.0" class="java.beans.XMLDecoder">  
 <object class="javax.swing.JFrame">  
 <void property="name">  
 <string>frame1</string>  
 </void>  
 <void property="bounds">  
 <object class="java.awt.Rectangle">  
 <int>0</int>  
 <int>0</int>  
 <int>200</int>  
 <int>200</int>  
 </object>  
 </void>  
 <void property="contentPane">  
 <void method="add">  
 <object class="javax.swing.JButton">  
 <void property="label">  
 <string>Hello</string>  
 </void>  
 </object>  
 </void>  
 </void>  
 <void property="visible">  
 <boolean>true</boolean>  
 </void>  
 </object>  
 </java>

The XML syntax uses the following conventions:

* Each element represents a method call.
* The "object" tag denotes an *expression* whose value is to be used as the argument to the enclosing element.
* The "void" tag denotes a *statement* which will be executed, but whose result will not be used as an argument to the enclosing method.
* Elements which contain elements use those elements as arguments, unless they have the tag: "void".
* The name of the method is denoted by the "method" attribute.
* XML's standard "id" and "idref" attributes are used to make references to previous expressions - so as to deal with circularities in the object graph.
* The "class" attribute is used to specify the target of a static method or constructor explicitly; its value being the fully qualified name of the class.
* Elements with the "void" tag are executed using the outer context as the target if no target is defined by a "class" attribute.
* Java's String class is treated specially and is written <string>Hello, world</string> where the characters of the string are converted to bytes using the UTF-8 character encoding.

Although all object graphs may be written using just these three tags, the following definitions are included so that common data structures can be expressed more concisely:

* The default method name is "new".
* A reference to a java class is written in the form <class>javax.swing.JButton</class>.
* Instances of the wrapper classes for Java's primitive types are written using the name of the primitive type as the tag. For example, an instance of the Integer class could be written: <int>123</int>. Note that the XMLEncoder class uses Java's reflection package in which the conversion between Java's primitive types and their associated "wrapper classes" is handled internally. The API for the XMLEncoder class itself deals only with Objects.
* In an element representing a nullary method whose name starts with "get", the "method" attribute is replaced with a "property" attribute whose value is given by removing the "get" prefix and decapitalizing the result.
* In an element representing a monadic method whose name starts with "set", the "method" attribute is replaced with a "property" attribute whose value is given by removing the "set" prefix and decapitalizing the result.
* In an element representing a method named "get" taking one integer argument, the "method" attribute is replaced with an "index" attribute whose value the value of the first argument.
* In an element representing a method named "set" taking two arguments, the first of which is an integer, the "method" attribute is replaced with an "index" attribute whose value the value of the first argument.
* A reference to an array is written using the "array" tag. The "class" and "length" attributes specify the sub-type of the array and its length respectively.

For more information you might also want to check out [Using XMLEncoder](http://java.sun.com/products/jfc/tsc/articles/persistence4), an article in *The Swing Connection.*

**Since:** 1.4 **See Also:**[XMLDecoder](http://docs.google.com/java/beans/XMLDecoder.html), [ObjectOutputStream](http://docs.google.com/java/io/ObjectOutputStream.html)

| **Constructor Summary** | |
| --- | --- |
| [**XMLEncoder**](http://docs.google.com/java/beans/XMLEncoder.html#XMLEncoder(java.io.OutputStream))([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)            Creates a new output stream for sending *JavaBeans* to the stream out using an XML encoding. |

| **Method Summary** | |
| --- | --- |
| void | [**close**](http://docs.google.com/java/beans/XMLEncoder.html#close())()            This method calls flush, writes the closing postamble and then closes the output stream associated with this stream. |
| void | [**flush**](http://docs.google.com/java/beans/XMLEncoder.html#flush())()            This method writes out the preamble associated with the XML encoding if it has not been written already and then writes out all of the values that been written to the stream since the last time flush was called. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getOwner**](http://docs.google.com/java/beans/XMLEncoder.html#getOwner())()            Gets the owner of this encoder. |
| void | [**setOwner**](http://docs.google.com/java/beans/XMLEncoder.html#setOwner(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) owner)            Sets the owner of this encoder to owner. |
| void | [**writeExpression**](http://docs.google.com/java/beans/XMLEncoder.html#writeExpression(java.beans.Expression))([Expression](http://docs.google.com/java/beans/Expression.html) oldExp)            Records the Expression so that the Encoder will produce the actual output when the stream is flushed. |
| void | [**writeObject**](http://docs.google.com/java/beans/XMLEncoder.html#writeObject(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Write an XML representation of the specified object to the output. |
| void | [**writeStatement**](http://docs.google.com/java/beans/XMLEncoder.html#writeStatement(java.beans.Statement))([Statement](http://docs.google.com/java/beans/Statement.html) oldStm)            Records the Statement so that the Encoder will produce the actual output when the stream is flushed. |

| **Methods inherited from class java.beans.**[**Encoder**](http://docs.google.com/java/beans/Encoder.html) |
| --- |
| [get](http://docs.google.com/java/beans/Encoder.html#get(java.lang.Object)), [getExceptionListener](http://docs.google.com/java/beans/Encoder.html#getExceptionListener()), [getPersistenceDelegate](http://docs.google.com/java/beans/Encoder.html#getPersistenceDelegate(java.lang.Class)), [remove](http://docs.google.com/java/beans/Encoder.html#remove(java.lang.Object)), [setExceptionListener](http://docs.google.com/java/beans/Encoder.html#setExceptionListener(java.beans.ExceptionListener)), [setPersistenceDelegate](http://docs.google.com/java/beans/Encoder.html#setPersistenceDelegate(java.lang.Class,%20java.beans.PersistenceDelegate)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### XMLEncoder

public **XMLEncoder**([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)

Creates a new output stream for sending *JavaBeans* to the stream out using an XML encoding.

**Parameters:**out - The stream to which the XML representation of the objects will be sent.**See Also:**[XMLDecoder.XMLDecoder(InputStream)](http://docs.google.com/java/beans/XMLDecoder.html#XMLDecoder(java.io.InputStream))

| **Method Detail** |
| --- |

### setOwner

public void **setOwner**([Object](http://docs.google.com/java/lang/Object.html) owner)

Sets the owner of this encoder to owner.

**Parameters:**owner - The owner of this encoder.**See Also:**[getOwner()](http://docs.google.com/java/beans/XMLEncoder.html#getOwner())

### getOwner

public [Object](http://docs.google.com/java/lang/Object.html) **getOwner**()

Gets the owner of this encoder.

**Returns:**The owner of this encoder.**See Also:**[setOwner(java.lang.Object)](http://docs.google.com/java/beans/XMLEncoder.html#setOwner(java.lang.Object))

### writeObject

public void **writeObject**([Object](http://docs.google.com/java/lang/Object.html) o)

Write an XML representation of the specified object to the output.

**Overrides:**[writeObject](http://docs.google.com/java/beans/Encoder.html#writeObject(java.lang.Object)) in class [Encoder](http://docs.google.com/java/beans/Encoder.html) **Parameters:**o - The object to be written to the stream.**See Also:**[XMLDecoder.readObject()](http://docs.google.com/java/beans/XMLDecoder.html#readObject())

### writeStatement

public void **writeStatement**([Statement](http://docs.google.com/java/beans/Statement.html) oldStm)

Records the Statement so that the Encoder will produce the actual output when the stream is flushed.

This method should only be invoked within the context of initializing a persistence delegate.

**Overrides:**[writeStatement](http://docs.google.com/java/beans/Encoder.html#writeStatement(java.beans.Statement)) in class [Encoder](http://docs.google.com/java/beans/Encoder.html) **Parameters:**oldStm - The statement that will be written to the stream.**See Also:**[PersistenceDelegate.initialize(java.lang.Class, java.lang.Object, java.lang.Object, java.beans.Encoder)](http://docs.google.com/java/beans/PersistenceDelegate.html#initialize(java.lang.Class,%20java.lang.Object,%20java.lang.Object,%20java.beans.Encoder))

### writeExpression

public void **writeExpression**([Expression](http://docs.google.com/java/beans/Expression.html) oldExp)

Records the Expression so that the Encoder will produce the actual output when the stream is flushed.

This method should only be invoked within the context of initializing a persistence delegate or setting up an encoder to read from a resource bundle.

For more information about using resource bundles with the XMLEncoder, see http://java.sun.com/products/jfc/tsc/articles/persistence4/#i18n

**Overrides:**[writeExpression](http://docs.google.com/java/beans/Encoder.html#writeExpression(java.beans.Expression)) in class [Encoder](http://docs.google.com/java/beans/Encoder.html) **Parameters:**oldExp - The expression that will be written to the stream.**See Also:**[PersistenceDelegate.initialize(java.lang.Class, java.lang.Object, java.lang.Object, java.beans.Encoder)](http://docs.google.com/java/beans/PersistenceDelegate.html#initialize(java.lang.Class,%20java.lang.Object,%20java.lang.Object,%20java.beans.Encoder))

### flush

public void **flush**()

This method writes out the preamble associated with the XML encoding if it has not been written already and then writes out all of the values that been written to the stream since the last time flush was called. After flushing, all internal references to the values that were written to this stream are cleared.

### close

public void **close**()

This method calls flush, writes the closing postamble and then closes the output stream associated with this stream.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/XMLEncoder.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/beans/XMLDecoder.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/beans/XMLEncoder.html)    [**NO FRAMES**](http://docs.google.com/XMLEncoder.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).