| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PersistenceDelegate.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/ParameterDescriptor.html)   [**NEXT CLASS**](http://docs.google.com/java/beans/PropertyChangeEvent.html) | [**FRAMES**](http://docs.google.com/index.html?java/beans/PersistenceDelegate.html)    [**NO FRAMES**](http://docs.google.com/PersistenceDelegate.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **java.beans**

Class PersistenceDelegate

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

**Direct Known Subclasses:** [DefaultPersistenceDelegate](http://docs.google.com/java/beans/DefaultPersistenceDelegate.html)

public abstract class **PersistenceDelegate**extends [Object](http://docs.google.com/java/lang/Object.html)

The PersistenceDelegate class takes the responsibility for expressing the state of an instance of a given class in terms of the methods in the class's public API. Instead of associating the responsibility of persistence with the class itself as is done, for example, by the readObject and writeObject methods used by the ObjectOutputStream, streams like the XMLEncoder which use this delegation model can have their behavior controlled independently of the classes themselves. Normally, the class is the best place to put such information and conventions can easily be expressed in this delegation scheme to do just that. Sometimes however, it is the case that a minor problem in a single class prevents an entire object graph from being written and this can leave the application developer with no recourse but to attempt to shadow the problematic classes locally or use alternative persistence techniques. In situations like these, the delegation model gives a relatively clean mechanism for the application developer to intervene in all parts of the serialization process without requiring that modifications be made to the implementation of classes which are not part of the application itself.

In addition to using a delegation model, this persistence scheme differs from traditional serialization schemes in requiring an analog of the writeObject method without a corresponding readObject method. The writeObject analog encodes each instance in terms of its public API and there is no need to define a readObject analog since the procedure for reading the serialized form is defined by the semantics of method invocation as laid out in the Java Language Specification. Breaking the dependency between writeObject and readObject implementations, which may change from version to version, is the key factor in making the archives produced by this technique immune to changes in the private implementations of the classes to which they refer.

A persistence delegate, may take control of all aspects of the persistence of an object including:

* Deciding whether or not an instance can be mutated into another instance of the same class.
* Instantiating the object, either by calling a public constructor or a public factory method.
* Performing the initialization of the object.

**Since:** 1.4 **See Also:**[XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html)

| **Constructor Summary** | |
| --- | --- |
| [**PersistenceDelegate**](http://docs.google.com/java/beans/PersistenceDelegate.html#PersistenceDelegate())() |

| **Method Summary** | |
| --- | --- |
| protected  void | [**initialize**](http://docs.google.com/java/beans/PersistenceDelegate.html#initialize(java.lang.Class,%20java.lang.Object,%20java.lang.Object,%20java.beans.Encoder))([Class](http://docs.google.com/java/lang/Class.html)<?> type, [Object](http://docs.google.com/java/lang/Object.html) oldInstance, [Object](http://docs.google.com/java/lang/Object.html) newInstance, [Encoder](http://docs.google.com/java/beans/Encoder.html) out)            Produce a series of statements with side effects on newInstance so that the new instance becomes *equivalent* to oldInstance. |
| protected abstract  [Expression](http://docs.google.com/java/beans/Expression.html) | [**instantiate**](http://docs.google.com/java/beans/PersistenceDelegate.html#instantiate(java.lang.Object,%20java.beans.Encoder))([Object](http://docs.google.com/java/lang/Object.html) oldInstance, [Encoder](http://docs.google.com/java/beans/Encoder.html) out)            Returns an expression whose value is oldInstance. |
| protected  boolean | [**mutatesTo**](http://docs.google.com/java/beans/PersistenceDelegate.html#mutatesTo(java.lang.Object,%20java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) oldInstance, [Object](http://docs.google.com/java/lang/Object.html) newInstance)            Returns true if an *equivalent* copy of oldInstance may be created by applying a series of statements to newInstance. |
| void | [**writeObject**](http://docs.google.com/java/beans/PersistenceDelegate.html#writeObject(java.lang.Object,%20java.beans.Encoder))([Object](http://docs.google.com/java/lang/Object.html) oldInstance, [Encoder](http://docs.google.com/java/beans/Encoder.html) out)            The writeObject is a single entry point to the persistence and is used by a Encoder in the traditional mode of delegation. |

| **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** |
| --- |

### PersistenceDelegate

public **PersistenceDelegate**()

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

### writeObject

public void **writeObject**([Object](http://docs.google.com/java/lang/Object.html) oldInstance,  
 [Encoder](http://docs.google.com/java/beans/Encoder.html) out)

The writeObject is a single entry point to the persistence and is used by a Encoder in the traditional mode of delegation. Although this method is not final, it should not need to be subclassed under normal circumstances.

This implementation first checks to see if the stream has already encountered this object. Next the mutatesTo method is called to see if that candidate returned from the stream can be mutated into an accurate copy of oldInstance. If it can, the initialize method is called to perform the initialization. If not, the candidate is removed from the stream, and the instantiate method is called to create a new candidate for this object.

**Parameters:**oldInstance - The instance that will be created by this expression.out - The stream to which this expression will be written.

### mutatesTo

protected boolean **mutatesTo**([Object](http://docs.google.com/java/lang/Object.html) oldInstance,  
 [Object](http://docs.google.com/java/lang/Object.html) newInstance)

Returns true if an *equivalent* copy of oldInstance may be created by applying a series of statements to newInstance. In the specification of this method, we mean by equivalent that the modified instance is indistinguishable from oldInstance in the behavior of the relevant methods in its public API. [Note: we use the phrase *relevant* methods rather than *all* methods here only because, to be strictly correct, methods like hashCode and toString prevent most classes from producing truly indistinguishable copies of their instances].

The default behavior returns true if the classes of the two instances are the same.

**Parameters:**oldInstance - The instance to be copied.newInstance - The instance that is to be modified. **Returns:**True if an equivalent copy of newInstance may be created by applying a series of mutations to oldInstance.

### instantiate

protected abstract [Expression](http://docs.google.com/java/beans/Expression.html) **instantiate**([Object](http://docs.google.com/java/lang/Object.html) oldInstance,  
 [Encoder](http://docs.google.com/java/beans/Encoder.html) out)

Returns an expression whose value is oldInstance. This method is used to characterize the constructor or factory method that should be used to create the given object. For example, the instantiate method of the persistence delegate for the Field class could be defined as follows:

Field f = (Field)oldInstance;  
 return new Expression(f, f.getDeclaringClass(), "getField", new Object[]{f.getName()});

Note that we declare the value of the returned expression so that the value of the expression (as returned by getValue) will be identical to oldInstance.

**Parameters:**oldInstance - The instance that will be created by this expression.out - The stream to which this expression will be written. **Returns:**An expression whose value is oldInstance.

### initialize

protected void **initialize**([Class](http://docs.google.com/java/lang/Class.html)<?> type,  
 [Object](http://docs.google.com/java/lang/Object.html) oldInstance,  
 [Object](http://docs.google.com/java/lang/Object.html) newInstance,  
 [Encoder](http://docs.google.com/java/beans/Encoder.html) out)

Produce a series of statements with side effects on newInstance so that the new instance becomes *equivalent* to oldInstance. In the specification of this method, we mean by equivalent that, after the method returns, the modified instance is indistinguishable from newInstance in the behavior of all methods in its public API.

The implementation typically achieves this goal by producing a series of "what happened" statements involving the oldInstance and its publicly available state. These statements are sent to the output stream using its writeExpression method which returns an expression involving elements in a cloned environment simulating the state of an input stream during reading. Each statement returned will have had all instances the old environment replaced with objects which exist in the new one. In particular, references to the target of these statements, which start out as references to oldInstance are returned as references to the newInstance instead. Executing these statements effects an incremental alignment of the state of the two objects as a series of modifications to the objects in the new environment. By the time the initialize method returns it should be impossible to tell the two instances apart by using their public APIs. Most importantly, the sequence of steps that were used to make these objects appear equivalent will have been recorded by the output stream and will form the actual output when the stream is flushed.

The default implementation, calls the initialize method of the type's superclass.

**Parameters:**oldInstance - The instance to be copied.newInstance - The instance that is to be modified.out - The stream to which any initialization statements should be written.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PersistenceDelegate.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/ParameterDescriptor.html)   [**NEXT CLASS**](http://docs.google.com/java/beans/PropertyChangeEvent.html) | [**FRAMES**](http://docs.google.com/index.html?java/beans/PersistenceDelegate.html)    [**NO FRAMES**](http://docs.google.com/PersistenceDelegate.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[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).