| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Properties.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/util/PriorityQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/PropertyPermission.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Properties.html)    [**NO FRAMES**](http://docs.google.com/Properties.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **java.util**

Class Properties

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.util.Dictionary](http://docs.google.com/java/util/Dictionary.html)<K,V>  
 [java.util.Hashtable](http://docs.google.com/java/util/Hashtable.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>  
 **java.util.Properties**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Map](http://docs.google.com/java/util/Map.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)> **Direct Known Subclasses:** [Provider](http://docs.google.com/java/security/Provider.html)

public class **Properties**extends [Hashtable](http://docs.google.com/java/util/Hashtable.html)<[Object](http://docs.google.com/java/lang/Object.html),[Object](http://docs.google.com/java/lang/Object.html)>

The Properties class represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string.

A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

Because Properties inherits from Hashtable, the put and putAll methods can be applied to a Properties object. Their use is strongly discouraged as they allow the caller to insert entries whose keys or values are not Strings. The setProperty method should be used instead. If the store or save method is called on a "compromised" Properties object that contains a non-String key or value, the call will fail. Similarly, the call to the propertyNames or list method will fail if it is called on a "compromised" Properties object that contains a non-String key.

The [load(Reader)](http://docs.google.com/java/util/Properties.html#load(java.io.Reader)) / [store(Writer, String)](http://docs.google.com/java/util/Properties.html#store(java.io.Writer,%20java.lang.String)) methods load and store properties from and to a character based stream in a simple line-oriented format specified below. The [load(InputStream)](http://docs.google.com/java/util/Properties.html#load(java.io.InputStream)) / [store(OutputStream, String)](http://docs.google.com/java/util/Properties.html#store(java.io.OutputStream,%20java.lang.String)) methods work the same way as the load(Reader)/store(Writer, String) pair, except the input/output stream is encoded in ISO 8859-1 character encoding. Characters that cannot be directly represented in this encoding can be written using [Unicode escapes](http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#3.3) ; only a single 'u' character is allowed in an escape sequence. The native2ascii tool can be used to convert property files to and from other character encodings.

The [loadFromXML(InputStream)](http://docs.google.com/java/util/Properties.html#loadFromXML(java.io.InputStream)) and [storeToXML(OutputStream, String, String)](http://docs.google.com/java/util/Properties.html#storeToXML(java.io.OutputStream,%20java.lang.String,%20java.lang.String)) methods load and store properties in a simple XML format. By default the UTF-8 character encoding is used, however a specific encoding may be specified if required. An XML properties document has the following DOCTYPE declaration:

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

Note that the system URI (http://java.sun.com/dtd/properties.dtd) is *not* accessed when exporting or importing properties; it merely serves as a string to uniquely identify the DTD, which is:

<?xml version="1.0" encoding="UTF-8"?>  
  
 <!-- DTD for properties -->  
  
 <!ELEMENT properties ( comment?, entry\* ) >  
  
 <!ATTLIST properties version CDATA #FIXED "1.0">  
  
 <!ELEMENT comment (#PCDATA) >  
  
 <!ELEMENT entry (#PCDATA) >  
  
 <!ATTLIST entry key CDATA #REQUIRED>

**Since:** JDK1.0 **See Also:**[native2ascii tool for Solaris](http://docs.google.com/technotes/tools/solaris/native2ascii.html), [native2ascii tool for Windows](http://docs.google.com/technotes/tools/windows/native2ascii.html)

This class is thread-safe: multiple threads can share a single Properties object without the need for external synchronization., [Serialized Form](http://docs.google.com/serialized-form.html#java.util.Properties)

| **Field Summary** | |
| --- | --- |
| protected  [Properties](http://docs.google.com/java/util/Properties.html) | [**defaults**](http://docs.google.com/java/util/Properties.html#defaults)            A property list that contains default values for any keys not found in this property list. |

| **Constructor Summary** | |
| --- | --- |
| [**Properties**](http://docs.google.com/java/util/Properties.html#Properties())()            Creates an empty property list with no default values. |
| [**Properties**](http://docs.google.com/java/util/Properties.html#Properties(java.util.Properties))([Properties](http://docs.google.com/java/util/Properties.html) defaults)            Creates an empty property list with the specified defaults. |

| **Method Summary** | |
| --- | --- |
| [String](http://docs.google.com/java/lang/String.html) | [**getProperty**](http://docs.google.com/java/util/Properties.html#getProperty(java.lang.String))([String](http://docs.google.com/java/lang/String.html) key)            Searches for the property with the specified key in this property list. |
| [String](http://docs.google.com/java/lang/String.html) | [**getProperty**](http://docs.google.com/java/util/Properties.html#getProperty(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) key, [String](http://docs.google.com/java/lang/String.html) defaultValue)            Searches for the property with the specified key in this property list. |
| void | [**list**](http://docs.google.com/java/util/Properties.html#list(java.io.PrintStream))([PrintStream](http://docs.google.com/java/io/PrintStream.html) out)            Prints this property list out to the specified output stream. |
| void | [**list**](http://docs.google.com/java/util/Properties.html#list(java.io.PrintWriter))([PrintWriter](http://docs.google.com/java/io/PrintWriter.html) out)            Prints this property list out to the specified output stream. |
| void | [**load**](http://docs.google.com/java/util/Properties.html#load(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) inStream)            Reads a property list (key and element pairs) from the input byte stream. |
| void | [**load**](http://docs.google.com/java/util/Properties.html#load(java.io.Reader))([Reader](http://docs.google.com/java/io/Reader.html) reader)            Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format. |
| void | [**loadFromXML**](http://docs.google.com/java/util/Properties.html#loadFromXML(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) in)            Loads all of the properties represented by the XML document on the specified input stream into this properties table. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html)<?> | [**propertyNames**](http://docs.google.com/java/util/Properties.html#propertyNames())()            Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list. |
| void | [**save**](http://docs.google.com/java/util/Properties.html#save(java.io.OutputStream,%20java.lang.String))([OutputStream](http://docs.google.com/java/io/OutputStream.html) out, [String](http://docs.google.com/java/lang/String.html) comments)  **Deprecated.** *This method does not throw an IOException if an I/O error occurs while saving the property list. The preferred way to save a properties list is via the store(OutputStream out, String comments) method or the storeToXML(OutputStream os, String comment) method.* |
| [Object](http://docs.google.com/java/lang/Object.html) | [**setProperty**](http://docs.google.com/java/util/Properties.html#setProperty(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) key, [String](http://docs.google.com/java/lang/String.html) value)            Calls the Hashtable method put. |
| void | [**store**](http://docs.google.com/java/util/Properties.html#store(java.io.OutputStream,%20java.lang.String))([OutputStream](http://docs.google.com/java/io/OutputStream.html) out, [String](http://docs.google.com/java/lang/String.html) comments)            Writes this property list (key and element pairs) in this Properties table to the output stream in a format suitable for loading into a Properties table using the [load(InputStream)](http://docs.google.com/java/util/Properties.html#load(java.io.InputStream)) method. |
| void | [**store**](http://docs.google.com/java/util/Properties.html#store(java.io.Writer,%20java.lang.String))([Writer](http://docs.google.com/java/io/Writer.html) writer, [String](http://docs.google.com/java/lang/String.html) comments)            Writes this property list (key and element pairs) in this Properties table to the output character stream in a format suitable for using the [load(Reader)](http://docs.google.com/java/util/Properties.html#load(java.io.Reader)) method. |
| void | [**storeToXML**](http://docs.google.com/java/util/Properties.html#storeToXML(java.io.OutputStream,%20java.lang.String))([OutputStream](http://docs.google.com/java/io/OutputStream.html) os, [String](http://docs.google.com/java/lang/String.html) comment)            Emits an XML document representing all of the properties contained in this table. |
| void | [**storeToXML**](http://docs.google.com/java/util/Properties.html#storeToXML(java.io.OutputStream,%20java.lang.String,%20java.lang.String))([OutputStream](http://docs.google.com/java/io/OutputStream.html) os, [String](http://docs.google.com/java/lang/String.html) comment, [String](http://docs.google.com/java/lang/String.html) encoding)            Emits an XML document representing all of the properties contained in this table, using the specified encoding. |
| [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> | [**stringPropertyNames**](http://docs.google.com/java/util/Properties.html#stringPropertyNames())()            Returns a set of keys in this property list where the key and its corresponding value are strings, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list. |

| **Methods inherited from class java.util.**[**Hashtable**](http://docs.google.com/java/util/Hashtable.html) |
| --- |
| [clear](http://docs.google.com/java/util/Hashtable.html#clear()), [clone](http://docs.google.com/java/util/Hashtable.html#clone()), [contains](http://docs.google.com/java/util/Hashtable.html#contains(java.lang.Object)), [containsKey](http://docs.google.com/java/util/Hashtable.html#containsKey(java.lang.Object)), [containsValue](http://docs.google.com/java/util/Hashtable.html#containsValue(java.lang.Object)), [elements](http://docs.google.com/java/util/Hashtable.html#elements()), [entrySet](http://docs.google.com/java/util/Hashtable.html#entrySet()), [equals](http://docs.google.com/java/util/Hashtable.html#equals(java.lang.Object)), [get](http://docs.google.com/java/util/Hashtable.html#get(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Hashtable.html#hashCode()), [isEmpty](http://docs.google.com/java/util/Hashtable.html#isEmpty()), [keys](http://docs.google.com/java/util/Hashtable.html#keys()), [keySet](http://docs.google.com/java/util/Hashtable.html#keySet()), [put](http://docs.google.com/java/util/Hashtable.html#put(K,%20V)), [putAll](http://docs.google.com/java/util/Hashtable.html#putAll(java.util.Map)), [rehash](http://docs.google.com/java/util/Hashtable.html#rehash()), [remove](http://docs.google.com/java/util/Hashtable.html#remove(java.lang.Object)), [size](http://docs.google.com/java/util/Hashtable.html#size()), [toString](http://docs.google.com/java/util/Hashtable.html#toString()), [values](http://docs.google.com/java/util/Hashtable.html#values()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [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)) |

| **Field Detail** |
| --- |

### defaults

protected [Properties](http://docs.google.com/java/util/Properties.html) **defaults**

A property list that contains default values for any keys not found in this property list.

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

### Properties

public **Properties**()

Creates an empty property list with no default values.

### Properties

public **Properties**([Properties](http://docs.google.com/java/util/Properties.html) defaults)

Creates an empty property list with the specified defaults.

**Parameters:**defaults - the defaults.

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

### setProperty

public [Object](http://docs.google.com/java/lang/Object.html) **setProperty**([String](http://docs.google.com/java/lang/String.html) key,  
 [String](http://docs.google.com/java/lang/String.html) value)

Calls the Hashtable method put. Provided for parallelism with the getProperty method. Enforces use of strings for property keys and values. The value returned is the result of the Hashtable call to put.

**Parameters:**key - the key to be placed into this property list.value - the value corresponding to key. **Returns:**the previous value of the specified key in this property list, or null if it did not have one.**Since:** 1.2 **See Also:**[getProperty(java.lang.String)](http://docs.google.com/java/util/Properties.html#getProperty(java.lang.String))

### load

public void **load**([Reader](http://docs.google.com/java/io/Reader.html) reader)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

Properties are processed in terms of lines. There are two kinds of line, *natural lines* and *logical lines*. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (\n or \r or \r\n) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' ', '\u0020'), tab ('\t', '\u0009'), and form feed ('\f', '\u000C') to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is *not* sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2*n* contiguous backslashes before a line terminator (or elsewhere) encodes *n* backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

\:\=

would be the two-character key ":=". Line terminator characters can be included using \r and \n escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

Truth = Beauty  
 Truth:Beauty  
 Truth :Beauty

As another example, the following three lines specify a single property:

fruits apple, banana, pear, \  
 cantaloupe, watermelon, \  
 kiwi, mango

The key is "fruits" and the associated element is:

"apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are *not* replaced by one or more other characters.

As a third example, the line:

cheeses

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see [§3.3](http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#3.3) and [§3.10.6](http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#3.10.6) of the *Java Language Specification*). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

* Octal escapes are not recognized.
* The character sequence \b does *not* represent a backspace character.
* The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a Java string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.
* Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.
* Only a single 'u' character is allowed in a Uniocde escape sequence.

The specified stream remains open after this method returns.

**Parameters:**reader - the input character stream. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurred when reading from the input stream. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if a malformed Unicode escape appears in the input.**Since:** 1.6

### load

public void **load**([InputStream](http://docs.google.com/java/io/InputStream.html) inStream)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reads a property list (key and element pairs) from the input byte stream. The input stream is in a simple line-oriented format as specified in [load(Reader)](http://docs.google.com/java/util/Properties.html#load(java.io.Reader)) and is assumed to use the ISO 8859-1 character encoding; that is each byte is one Latin1 character. Characters not in Latin1, and certain special characters, are represented in keys and elements using [Unicode escapes](http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#3.3).

The specified stream remains open after this method returns.

**Parameters:**inStream - the input stream. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an error occurred when reading from the input stream. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the input stream contains a malformed Unicode escape sequence.**Since:** 1.2

### save

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **save**([OutputStream](http://docs.google.com/java/io/OutputStream.html) out,  
 [String](http://docs.google.com/java/lang/String.html) comments)

**Deprecated.** *This method does not throw an IOException if an I/O error occurs while saving the property list. The preferred way to save a properties list is via the store(OutputStream out, String comments) method or the storeToXML(OutputStream os, String comment) method.*

Calls the store(OutputStream out, String comments) method and suppresses IOExceptions that were thrown.

**Parameters:**out - an output stream.comments - a description of the property list. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Properties object contains any keys or values that are not Strings.

### store

public void **store**([Writer](http://docs.google.com/java/io/Writer.html) writer,  
 [String](http://docs.google.com/java/lang/String.html) comments)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes this property list (key and element pairs) in this Properties table to the output character stream in a format suitable for using the [load(Reader)](http://docs.google.com/java/util/Properties.html#load(java.io.Reader)) method.

Properties from the defaults table of this Properties table (if any) are *not* written out by this method.

If the comments argument is not null, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of Date for the current time), and a line separator as generated by the Writer.

Then every entry in this Properties table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

**Parameters:**writer - an output character stream writer.comments - a description of the property list. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing this property list to the specified output stream throws an IOException. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Properties object contains any keys or values that are not Strings. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if writer is null.**Since:** 1.6

### store

public void **store**([OutputStream](http://docs.google.com/java/io/OutputStream.html) out,  
 [String](http://docs.google.com/java/lang/String.html) comments)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes this property list (key and element pairs) in this Properties table to the output stream in a format suitable for loading into a Properties table using the [load(InputStream)](http://docs.google.com/java/util/Properties.html#load(java.io.InputStream)) method.

Properties from the defaults table of this Properties table (if any) are *not* written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in [store(Writer)](http://docs.google.com/java/util/Properties.html#store(java.io.Writer,%20java.lang.String)), with the following differences:

* The stream is written using the ISO 8859-1 character encoding.
* Characters not in Latin-1 in the comments are written as \u*xxxx* for their appropriate unicode hexadecimal value *xxxx*.
* Characters less than \u0020 and characters greater than \u007E in property keys or values are written as \u*xxxx* for the appropriate hexadecimal value *xxxx*.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

**Parameters:**out - an output stream.comments - a description of the property list. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing this property list to the specified output stream throws an IOException. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Properties object contains any keys or values that are not Strings. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if out is null.**Since:** 1.2

### loadFromXML

public void **loadFromXML**([InputStream](http://docs.google.com/java/io/InputStream.html) in)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [InvalidPropertiesFormatException](http://docs.google.com/java/util/InvalidPropertiesFormatException.html)

Loads all of the properties represented by the XML document on the specified input stream into this properties table.

The XML document must have the following DOCTYPE declaration:

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

Furthermore, the document must satisfy the properties DTD described above.

The specified stream is closed after this method returns.

**Parameters:**in - the input stream from which to read the XML document. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if reading from the specified input stream results in an IOException. [InvalidPropertiesFormatException](http://docs.google.com/java/util/InvalidPropertiesFormatException.html) - Data on input stream does not constitute a valid XML document with the mandated document type. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if in is null.**Since:** 1.5 **See Also:**[storeToXML(OutputStream, String, String)](http://docs.google.com/java/util/Properties.html#storeToXML(java.io.OutputStream,%20java.lang.String,%20java.lang.String))

### storeToXML

public void **storeToXML**([OutputStream](http://docs.google.com/java/io/OutputStream.html) os,  
 [String](http://docs.google.com/java/lang/String.html) comment)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Emits an XML document representing all of the properties contained in this table.

An invocation of this method of the form props.storeToXML(os, comment) behaves in exactly the same way as the invocation props.storeToXML(os, comment, "UTF-8");.

**Parameters:**os - the output stream on which to emit the XML document.comment - a description of the property list, or null if no comment is desired. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing to the specified output stream results in an IOException. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if os is null. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Properties object contains any keys or values that are not Strings.**Since:** 1.5 **See Also:**[loadFromXML(InputStream)](http://docs.google.com/java/util/Properties.html#loadFromXML(java.io.InputStream))

### storeToXML

public void **storeToXML**([OutputStream](http://docs.google.com/java/io/OutputStream.html) os,  
 [String](http://docs.google.com/java/lang/String.html) comment,  
 [String](http://docs.google.com/java/lang/String.html) encoding)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Emits an XML document representing all of the properties contained in this table, using the specified encoding.

The XML document will have the following DOCTYPE declaration:

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

If the specified comment is null then no comment will be stored in the document.

The specified stream remains open after this method returns.

**Parameters:**os - the output stream on which to emit the XML document.comment - a description of the property list, or null if no comment is desired. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing to the specified output stream results in an IOException. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if os is null, or if encoding is null. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if this Properties object contains any keys or values that are not Strings.**Since:** 1.5 **See Also:**[loadFromXML(InputStream)](http://docs.google.com/java/util/Properties.html#loadFromXML(java.io.InputStream))

### getProperty

public [String](http://docs.google.com/java/lang/String.html) **getProperty**([String](http://docs.google.com/java/lang/String.html) key)

Searches for the property with the specified key in this property list. If the key is not found in this property list, the default property list, and its defaults, recursively, are then checked. The method returns null if the property is not found.

**Parameters:**key - the property key. **Returns:**the value in this property list with the specified key value.**See Also:**[setProperty(java.lang.String, java.lang.String)](http://docs.google.com/java/util/Properties.html#setProperty(java.lang.String,%20java.lang.String)), [defaults](http://docs.google.com/java/util/Properties.html#defaults)

### getProperty

public [String](http://docs.google.com/java/lang/String.html) **getProperty**([String](http://docs.google.com/java/lang/String.html) key,  
 [String](http://docs.google.com/java/lang/String.html) defaultValue)

Searches for the property with the specified key in this property list. If the key is not found in this property list, the default property list, and its defaults, recursively, are then checked. The method returns the default value argument if the property is not found.

**Parameters:**key - the hashtable key.defaultValue - a default value. **Returns:**the value in this property list with the specified key value.**See Also:**[setProperty(java.lang.String, java.lang.String)](http://docs.google.com/java/util/Properties.html#setProperty(java.lang.String,%20java.lang.String)), [defaults](http://docs.google.com/java/util/Properties.html#defaults)

### propertyNames

public [Enumeration](http://docs.google.com/java/util/Enumeration.html)<?> **propertyNames**()

Returns an enumeration of all the keys in this property list, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list.

**Returns:**an enumeration of all the keys in this property list, including the keys in the default property list. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if any key in this property list is not a string.**See Also:**[Enumeration](http://docs.google.com/java/util/Enumeration.html), [defaults](http://docs.google.com/java/util/Properties.html#defaults), [stringPropertyNames()](http://docs.google.com/java/util/Properties.html#stringPropertyNames())

### stringPropertyNames

public [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> **stringPropertyNames**()

Returns a set of keys in this property list where the key and its corresponding value are strings, including distinct keys in the default property list if a key of the same name has not already been found from the main properties list. Properties whose key or value is not of type String are omitted.

The returned set is not backed by the Properties object. Changes to this Properties are not reflected in the set, or vice versa.

**Returns:**a set of keys in this property list where the key and its corresponding value are strings, including the keys in the default property list.**Since:** 1.6 **See Also:**[defaults](http://docs.google.com/java/util/Properties.html#defaults)

### list

public void **list**([PrintStream](http://docs.google.com/java/io/PrintStream.html) out)

Prints this property list out to the specified output stream. This method is useful for debugging.

**Parameters:**out - an output stream. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if any key in this property list is not a string.

### list

public void **list**([PrintWriter](http://docs.google.com/java/io/PrintWriter.html) out)

Prints this property list out to the specified output stream. This method is useful for debugging.

**Parameters:**out - an output stream. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if any key in this property list is not a string.**Since:** JDK1.1

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Properties.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/util/PriorityQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/PropertyPermission.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Properties.html)    [**NO FRAMES**](http://docs.google.com/Properties.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[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).