| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Filer.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/javax/annotation/processing/Completions.html)   [**NEXT CLASS**](http://docs.google.com/javax/annotation/processing/FilerException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/annotation/processing/Filer.html)    [**NO FRAMES**](http://docs.google.com/Filer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.annotation.processing**

Interface Filer

public interface **Filer**

This interface supports the creation of new files by an annotation processor. Files created in this way will be known to the annotation processing tool implementing this interface, better enabling the tool to manage them. Source and class files so created will be considered for processing by the tool after the close method has been called on the Writer or OutputStream used to write the contents of the file. Three kinds of files are distinguished: source files, class files, and auxiliary resource files.

There are two distinguished supported locations (subtrees within the logical file system) where newly created files are placed: one for [new source files](http://docs.google.com/javax/tools/StandardLocation.html#SOURCE_OUTPUT), and one for [new class files](http://docs.google.com/javax/tools/StandardLocation.html#CLASS_OUTPUT). (These might be specified on a tool's command line, for example, using flags such as -s and -d.) The actual locations for new source files and new class files may or may not be distinct on a particular run of the tool. Resource files may be created in either location. The methods for reading and writing resources take a relative name argument. A relative name is a non-null, non-empty sequence of path segments separated by '/'; '.' and '..' are invalid path segments. A valid relative name must match the "path-rootless" rule of [RFC 3986](http://www.ietf.org/rfc/rfc3986.txt), section 3.3.

The file creation methods take a variable number of arguments to allow the *originating elements* to be provided as hints to the tool infrastructure to better manage dependencies. The originating elements are the types or packages (representing package-info files) which caused an annotation processor to attempt to create a new file. For example, if an annotation processor tries to create a source file, GeneratedFromUserSource, in response to processing

@Generate  
 public class UserSource {}

the type element for UserSource should be passed as part of the creation method call as in:

filer.createSourceFile("GeneratedFromUserSource",  
 eltUtils.getTypeElement("UserSource"));

If there are no originating elements, none need to be passed. This information may be used in an incremental environment to determine the need to rerun processors or remove generated files. Non-incremental environments may ignore the originating element information.

During each run of an annotation processing tool, a file with a given pathname may be created only once. If that file already exists before the first attempt to create it, the old contents will be deleted. Any subsequent attempt to create the same file during a run will throw a [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html), as will attempting to create both a class file and source file for the same type name or same package name. The [initial inputs](http://docs.google.com/javax/annotation/processing/Processor.html) to the tool are considered to be created by the zeroth round; therefore, attempting to create a source or class file corresponding to one of those inputs will result in a [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html).

In general, processors must not knowingly attempt to overwrite existing files that were not generated by some processor. A Filer may reject attempts to open a file corresponding to an existing type, like java.lang.Object. Likewise, the invoker of the annotation processing tool must not knowingly configure the tool such that the discovered processors will attempt to overwrite existing files that were not generated.

Processors can indicate a source or class file is generated by including an [@Generated](http://docs.google.com/javax/annotation/Generated.html) annotation.

Note that some of the effect of overwriting a file can be achieved by using a *decorator*-style pattern. Instead of modifying a class directly, the class is designed so that either its superclass is generated by annotation processing or subclasses of the class are generated by annotation processing. If the subclasses are generated, the parent class may be designed to use factories instead of public constructors so that only subclass instances would be presented to clients of the parent class.

**Since:** 1.6

| **Method Summary** | |
| --- | --- |
| [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) | [**createClassFile**](http://docs.google.com/javax/annotation/processing/Filer.html#createClassFile(java.lang.CharSequence,%20javax.lang.model.element.Element...))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) name, [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)            Creates a new class file, and returns an object to allow writing to it. |
| [FileObject](http://docs.google.com/javax/tools/FileObject.html) | [**createResource**](http://docs.google.com/javax/annotation/processing/Filer.html#createResource(javax.tools.JavaFileManager.Location,%20java.lang.CharSequence,%20java.lang.CharSequence,%20javax.lang.model.element.Element...))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [CharSequence](http://docs.google.com/java/lang/CharSequence.html) pkg, [CharSequence](http://docs.google.com/java/lang/CharSequence.html) relativeName, [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)            Creates a new auxiliary resource file for writing and returns a file object for it. |
| [JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) | [**createSourceFile**](http://docs.google.com/javax/annotation/processing/Filer.html#createSourceFile(java.lang.CharSequence,%20javax.lang.model.element.Element...))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) name, [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)            Creates a new source file and returns an object to allow writing to it. |
| [FileObject](http://docs.google.com/javax/tools/FileObject.html) | [**getResource**](http://docs.google.com/javax/annotation/processing/Filer.html#getResource(javax.tools.JavaFileManager.Location,%20java.lang.CharSequence,%20java.lang.CharSequence))([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location, [CharSequence](http://docs.google.com/java/lang/CharSequence.html) pkg, [CharSequence](http://docs.google.com/java/lang/CharSequence.html) relativeName)            Returns an object for reading an existing resource. |

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

### createSourceFile

[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) **createSourceFile**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) name,  
 [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a new source file and returns an object to allow writing to it. The file's name and path (relative to the [root output location for source files](http://docs.google.com/javax/tools/StandardLocation.html#SOURCE_OUTPUT)) are based on the type to be declared in that file. If more than one type is being declared, the name of the principal top-level type (the public one, for example) should be used. A source file can also be created to hold information about a package, including package annotations. To create a source file for a named package, have name be the package's name followed by ".package-info"; to create a source file for an unnamed package, use "package-info".

Note that to use a particular [charset](http://docs.google.com/java/nio/charset/Charset.html) to encode the contents of the file, an OutputStreamWriter with the chosen charset can be created from the OutputStream from the returned object. If the Writer from the returned object is directly used for writing, its charset is determined by the implementation. An annotation processing tool may have an -encoding flag or analogous option for specifying this; otherwise, it will typically be the platform's default encoding.

To avoid subsequent errors, the contents of the source file should be compatible with the [source version](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html#getSourceVersion()) being used for this run.

**Parameters:**name - canonical (fully qualified) name of the principal type being declared in this file or a package name followed by ".package-info" for a package information fileoriginatingElements - type or package elements causally associated with the creation of this file, may be elided or null **Returns:**a JavaFileObject to write the new source file **Throws:** [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html) - if the same pathname has already been created, the same type has already been created, or the name is not valid for a type [IOException](http://docs.google.com/java/io/IOException.html) - if the file cannot be created

### createClassFile

[JavaFileObject](http://docs.google.com/javax/tools/JavaFileObject.html) **createClassFile**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) name,  
 [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a new class file, and returns an object to allow writing to it. The file's name and path (relative to the [root output location for class files](http://docs.google.com/javax/tools/StandardLocation.html#CLASS_OUTPUT)) are based on the name of the type being written. A class file can also be created to hold information about a package, including package annotations. To create a class file for a named package, have name be the package's name followed by ".package-info"; creating a class file for an unnamed package is not supported.

To avoid subsequent errors, the contents of the class file should be compatible with the [source version](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html#getSourceVersion()) being used for this run.

**Parameters:**name - binary name of the type being written or a package name followed by ".package-info" for a package information fileoriginatingElements - type or package elements causally associated with the creation of this file, may be elided or null **Returns:**a JavaFileObject to write the new class file **Throws:** [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html) - if the same pathname has already been created, the same type has already been created, or the name is not valid for a type [IOException](http://docs.google.com/java/io/IOException.html) - if the file cannot be created

### createResource

[FileObject](http://docs.google.com/javax/tools/FileObject.html) **createResource**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [CharSequence](http://docs.google.com/java/lang/CharSequence.html) pkg,  
 [CharSequence](http://docs.google.com/java/lang/CharSequence.html) relativeName,  
 [Element](http://docs.google.com/javax/lang/model/element/Element.html)... originatingElements)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates a new auxiliary resource file for writing and returns a file object for it. The file may be located along with the newly created source files, newly created binary files, or other supported location. The locations [CLASS\_OUTPUT](http://docs.google.com/javax/tools/StandardLocation.html#CLASS_OUTPUT) and [SOURCE\_OUTPUT](http://docs.google.com/javax/tools/StandardLocation.html#SOURCE_OUTPUT) must be supported. The resource may be named relative to some package (as are source and class files), and from there by a relative pathname. In a loose sense, the full pathname of the new file will be the concatenation of location, pkg, and relativeName.

Files created via this method are not registered for annotation processing, even if the full pathname of the file would correspond to the full pathname of a new source file or new class file.

**Parameters:**location - location of the new filepkg - package relative to which the file should be named, or the empty string if nonerelativeName - final pathname components of the fileoriginatingElements - type or package elements causally associated with the creation of this file, may be elided or null **Returns:**a FileObject to write the new resource **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if the file cannot be created [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html) - if the same pathname has already been created [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an unsupported location [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if relativeName is not relative

### getResource

[FileObject](http://docs.google.com/javax/tools/FileObject.html) **getResource**([JavaFileManager.Location](http://docs.google.com/javax/tools/JavaFileManager.Location.html) location,  
 [CharSequence](http://docs.google.com/java/lang/CharSequence.html) pkg,  
 [CharSequence](http://docs.google.com/java/lang/CharSequence.html) relativeName)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Returns an object for reading an existing resource. The locations [CLASS\_OUTPUT](http://docs.google.com/javax/tools/StandardLocation.html#CLASS_OUTPUT) and [SOURCE\_OUTPUT](http://docs.google.com/javax/tools/StandardLocation.html#SOURCE_OUTPUT) must be supported.

**Parameters:**location - location of the filepkg - package relative to which the file should be searched, or the empty string if nonerelativeName - final pathname components of the file **Returns:**an object to read the file **Throws:** [FilerException](http://docs.google.com/javax/annotation/processing/FilerException.html) - if the same pathname has already been opened for writing [IOException](http://docs.google.com/java/io/IOException.html) - if the file cannot be opened [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an unsupported location [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if relativeName is not relative

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Filer.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/javax/annotation/processing/Completions.html)   [**NEXT CLASS**](http://docs.google.com/javax/annotation/processing/FilerException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/annotation/processing/Filer.html)    [**NO FRAMES**](http://docs.google.com/Filer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[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).