| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Processor.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/ProcessingEnvironment.html)   [**NEXT CLASS**](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html) | [**FRAMES**](http://docs.google.com/index.html?javax/annotation/processing/Processor.html)    [**NO FRAMES**](http://docs.google.com/Processor.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 Processor

**All Known Implementing Classes:** [AbstractProcessor](http://docs.google.com/javax/annotation/processing/AbstractProcessor.html)

public interface **Processor**

The interface for an annotation processor.

Annotation processing happens in a sequence of [rounds](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html). On each round, a processor may be asked to [process](http://docs.google.com/javax/annotation/processing/Processor.html#process(java.util.Set,%20javax.annotation.processing.RoundEnvironment)) a subset of the annotations found on the source and class files produced by a prior round. The inputs to the first round of processing are the initial inputs to a run of the tool; these initial inputs can be regarded as the output of a virtual zeroth round of processing. If a processor was asked to process on a given round, it will be asked to process on subsequent rounds, including the last round, even if there are no annotations for it to process. The tool infrastructure may also ask a processor to process files generated implicitly by the tool's operation.

Each implementation of a Processor must provide a public no-argument constructor to be used by tools to instantiate the processor. The tool infrastructure will interact with classes implementing this interface as follows:

1. If an existing Processor object is not being used, to create an instance of a processor the tool calls the no-arg constructor of the processor class.
2. Next, the tool calls the [init](http://docs.google.com/javax/annotation/processing/Processor.html#init(javax.annotation.processing.ProcessingEnvironment)) method with an appropriate ProcessingEnvironment.
3. Afterwards, the tool calls [getSupportedAnnotationTypes](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedAnnotationTypes()), [getSupportedOptions](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedOptions()), and [getSupportedSourceVersion](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedSourceVersion()). These methods are only called once per run, not on each round.
4. As appropriate, the tool calls the [process](http://docs.google.com/javax/annotation/processing/Processor.html#process(java.util.Set,%20javax.annotation.processing.RoundEnvironment)) method on the Processor object; a new Processor object is *not* created for each round.

If a processor object is created and used without the above protocol being followed, then the processor's behavior is not defined by this interface specification.

The tool uses a *discovery process* to find annotation processors and decide whether or not they should be run. By configuring the tool, the set of potential processors can be controlled. For example, for a [JavaCompiler](http://docs.google.com/javax/tools/JavaCompiler.html) the list of candidate processors to run can be [set directly](http://docs.google.com/javax/tools/JavaCompiler.CompilationTask.html#setProcessors(java.lang.Iterable)) or controlled by a [search path](http://docs.google.com/javax/tools/StandardLocation.html#ANNOTATION_PROCESSOR_PATH) used for a [service-style](http://docs.google.com/java/util/ServiceLoader.html) lookup. Other tool implementations may have different configuration mechanisms, such as command line options; for details, refer to the particular tool's documentation. Which processors the tool asks to [run](http://docs.google.com/javax/annotation/processing/Processor.html#process(java.util.Set,%20javax.annotation.processing.RoundEnvironment)) is a function of what annotations are present on the [root elements](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html#getRootElements()), what [annotation types a processor processes](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedAnnotationTypes()), and whether or not a processor [claims the annotations it processes](http://docs.google.com/javax/annotation/processing/Processor.html#process(java.util.Set,%20javax.annotation.processing.RoundEnvironment)). A processor will be asked to process a subset of the annotation types it supports, possibly an empty set. For a given round, the tool computes the set of annotation types on the root elements. If there is at least one annotation type present, as processors claim annotation types, they are removed from the set of unmatched annotations. When the set is empty or no more processors are available, the round has run to completion. If there are no annotation types present, annotation processing still occurs but only *universal processors* which support processing "\*" can claim the (empty) set of annotation types.

Note that if a processor supports "\*" and returns true, all annotations are claimed. Therefore, a universal processor being used to, for example, implement additional validity checks should return false so as to not prevent other such checkers from being able to run.

If a processor throws an uncaught exception, the tool may cease other active annotation processors. If a processor raises an error, the current round will run to completion and the subsequent round will indicate an [error was raised](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html#errorRaised()). Since annotation processors are run in a cooperative environment, a processor should throw an uncaught exception only in situations where no error recovery or reporting is feasible.

The tool environment is not required to support annotation processors that access environmental resources, either [per round](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html) or [cross-round](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html), in a multi-threaded fashion.

If the methods that return configuration information about the annotation processor return null, return other invalid input, or throw an exception, the tool infrastructure must treat this as an error condition.

To be robust when running in different tool implementations, an annotation processor should have the following properties:

1. The result of processing a given input is not a function of the presence or absence of other inputs (orthogonality).
2. Processing the same input produces the same output (consistency).
3. Processing input *A* followed by processing input *B* is equivalent to processing *B* then *A* (commutativity)
4. Processing an input does not rely on the presence of the output of other annotation processors (independence)

The [Filer](http://docs.google.com/javax/annotation/processing/Filer.html) interface discusses restrictions on how processors can operate on files.

Note that implementors of this interface may find it convenient to extend [AbstractProcessor](http://docs.google.com/javax/annotation/processing/AbstractProcessor.html) rather than implementing this interface directly.

**Since:** 1.6

| **Method Summary** | |
| --- | --- |
| [Iterable](http://docs.google.com/java/lang/Iterable.html)<? extends [Completion](http://docs.google.com/javax/annotation/processing/Completion.html)> | [**getCompletions**](http://docs.google.com/javax/annotation/processing/Processor.html#getCompletions(javax.lang.model.element.Element,%20javax.lang.model.element.AnnotationMirror,%20javax.lang.model.element.ExecutableElement,%20java.lang.String))([Element](http://docs.google.com/javax/lang/model/element/Element.html) element, [AnnotationMirror](http://docs.google.com/javax/lang/model/element/AnnotationMirror.html) annotation, [ExecutableElement](http://docs.google.com/javax/lang/model/element/ExecutableElement.html) member, [String](http://docs.google.com/java/lang/String.html) userText)            Returns to the tool infrastructure an iterable of suggested completions to an annotation. |
| [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getSupportedAnnotationTypes**](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedAnnotationTypes())()            Returns the names of the annotation types supported by this processor. |
| [Set](http://docs.google.com/java/util/Set.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getSupportedOptions**](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedOptions())()            Returns the options recognized by this processor. |
| [SourceVersion](http://docs.google.com/javax/lang/model/SourceVersion.html) | [**getSupportedSourceVersion**](http://docs.google.com/javax/annotation/processing/Processor.html#getSupportedSourceVersion())()            Returns the latest source version supported by this annotation processor. |
| void | [**init**](http://docs.google.com/javax/annotation/processing/Processor.html#init(javax.annotation.processing.ProcessingEnvironment))([ProcessingEnvironment](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html) processingEnv)            Initializes the processor with the processing environment. |
| boolean | [**process**](http://docs.google.com/javax/annotation/processing/Processor.html#process(java.util.Set,%20javax.annotation.processing.RoundEnvironment))([Set](http://docs.google.com/java/util/Set.html)<? extends [TypeElement](http://docs.google.com/javax/lang/model/element/TypeElement.html)> annotations, [RoundEnvironment](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html) roundEnv)            Processes a set of annotation types on type elements originating from the prior round and returns whether or not these annotations are claimed by this processor. |

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

### getSupportedOptions

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

Returns the options recognized by this processor. An implementation of the processing tool must provide a way to pass processor-specific options distinctly from options passed to the tool itself, see [getOptions](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html#getOptions()).

Each string returned in the set must be a period separated sequence of [identifiers](http://docs.google.com/javax/lang/model/SourceVersion.html#isIdentifier(java.lang.CharSequence)):

*SupportedOptionString:* *Identifiers*

*Identifiers:* *Identifier* *Identifier* . *Identifiers*

*Identifier:* Syntactic identifier, including keywords and literals

A tool might use this information to determine if any options provided by a user are unrecognized by any processor, in which case it may wish to report a warning.

**Returns:**the options recognized by this processor or an empty collection if none**See Also:**[SupportedOptions](http://docs.google.com/javax/annotation/processing/SupportedOptions.html)

### getSupportedAnnotationTypes

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

Returns the names of the annotation types supported by this processor. An element of the result may be the canonical (fully qualified) name of a supported annotation type. Alternately it may be of the form "*name*.\*" representing the set of all annotation types with canonical names beginning with "*name.*". Finally, "\*" by itself represents the set of all annotation types, including the empty set. Note that a processor should not claim "\*" unless it is actually processing all files; claiming unnecessary annotations may cause a performance slowdown in some environments.

Each string returned in the set must be accepted by the following grammar:

*SupportedAnnotationTypeString:* *TypeName* *DotStaropt* \*

*DotStar:* . \*where *TypeName* is as defined in the *Java Language Specification*.

**Returns:**the names of the annotation types supported by this processor**See Also:**[SupportedAnnotationTypes](http://docs.google.com/javax/annotation/processing/SupportedAnnotationTypes.html)**See**  [**The Java Language Specification, Third Edition**](http://java.sun.com/docs/books/jls/)**:** 3.8 Identifiers, 6.5.5 Meaning of Type Names

### getSupportedSourceVersion

[SourceVersion](http://docs.google.com/javax/lang/model/SourceVersion.html) **getSupportedSourceVersion**()

Returns the latest source version supported by this annotation processor.

**Returns:**the latest source version supported by this annotation processor.**See Also:**[SupportedSourceVersion](http://docs.google.com/javax/annotation/processing/SupportedSourceVersion.html), [ProcessingEnvironment.getSourceVersion()](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html#getSourceVersion())

### init

void **init**([ProcessingEnvironment](http://docs.google.com/javax/annotation/processing/ProcessingEnvironment.html) processingEnv)

Initializes the processor with the processing environment.

**Parameters:**processingEnv - environment for facilities the tool framework provides to the processor

### process

boolean **process**([Set](http://docs.google.com/java/util/Set.html)<? extends [TypeElement](http://docs.google.com/javax/lang/model/element/TypeElement.html)> annotations,  
 [RoundEnvironment](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html) roundEnv)

Processes a set of annotation types on type elements originating from the prior round and returns whether or not these annotations are claimed by this processor. If true is returned, the annotations are claimed and subsequent processors will not be asked to process them; if false is returned, the annotations are unclaimed and subsequent processors may be asked to process them. A processor may always return the same boolean value or may vary the result based on chosen criteria.

The input set will be empty if the processor supports "\*" and the root elements have no annotations. A Processor must gracefully handle an empty set of annotations.

**Parameters:**annotations - the annotation types requested to be processedroundEnv - environment for information about the current and prior round **Returns:**whether or not the set of annotations are claimed by this processor

### getCompletions

[Iterable](http://docs.google.com/java/lang/Iterable.html)<? extends [Completion](http://docs.google.com/javax/annotation/processing/Completion.html)> **getCompletions**([Element](http://docs.google.com/javax/lang/model/element/Element.html) element,  
 [AnnotationMirror](http://docs.google.com/javax/lang/model/element/AnnotationMirror.html) annotation,  
 [ExecutableElement](http://docs.google.com/javax/lang/model/element/ExecutableElement.html) member,  
 [String](http://docs.google.com/java/lang/String.html) userText)

Returns to the tool infrastructure an iterable of suggested completions to an annotation. Since completions are being asked for, the information provided about the annotation may be incomplete, as if for a source code fragment. A processor may return an empty iterable. Annotation processors should focus their efforts on providing completions for annotation members with additional validity constraints known to the processor, for example an int member whose value should lie between 1 and 10 or a string member that should be recognized by a known grammar, such as a regular expression or a URL.

Since incomplete programs are being modeled, some of the parameters may only have partial information or may be null. At least one of element and userText must be non-null. If element is non-null, annotation and member may be null. Processors may not throw a NullPointerException if some parameters are null; if a processor has no completions to offer based on the provided information, an empty iterable can be returned. The processor may also return a single completion with an empty value string and a message describing why there are no completions.

Completions are informative and may reflect additional validity checks performed by annotation processors. For example, consider the simple annotation:

@MersennePrime {  
 int value();  
 }

(A Mersenne prime is prime number of the form 2*n* - 1.) Given an AnnotationMirror for this annotation type, a list of all such primes in the int range could be returned without examining any other arguments to getCompletions:

import static javax.annotation.processing.Completions.\*;  
 ...  
 return Arrays.asList([of](http://docs.google.com/javax/annotation/processing/Completions.html#of(java.lang.String))("3"),  
 of("7"),  
 of("31"),  
 of("127"),  
 of("8191"),  
 of("131071"),  
 of("524287"),  
 of("2147483647"));

A more informative set of completions would include the number of each prime:

return Arrays.asList([of](http://docs.google.com/javax/annotation/processing/Completions.html#of(java.lang.String,%20java.lang.String))("3", "M2"),  
 of("7", "M3"),  
 of("31", "M5"),  
 of("127", "M7"),  
 of("8191", "M13"),  
 of("131071", "M17"),  
 of("524287", "M19"),  
 of("2147483647", "M31"));

However, if the userText is available, it can be checked to see if only a subset of the Mersenne primes are valid. For example, if the user has typed@MersennePrime(1the value of userText will be "1"; and only two of the primes are possible completions:

return Arrays.asList(of("127", "M7"),  
 of("131071", "M17"));

Sometimes no valid completion is possible. For example, there is no in-range Mersenne prime starting with 9:@MersennePrime(9An appropriate response in this case is to either return an empty list of completions,

return Collections.emptyList();

or a single empty completion with a helpful message

return Arrays.asList(of("", "No in-range Mersenne primes start with 9"));

**Parameters:**element - the element being annotatedannotation - the (perhaps partial) annotation being applied to the elementmember - the annotation member to return possible completions foruserText - source code text to be completed **Returns:**suggested completions to the annotation

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Processor.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/ProcessingEnvironment.html)   [**NEXT CLASS**](http://docs.google.com/javax/annotation/processing/RoundEnvironment.html) | [**FRAMES**](http://docs.google.com/index.html?javax/annotation/processing/Processor.html)    [**NO FRAMES**](http://docs.google.com/Processor.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).