LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-402071

Java Metadata Facility

D. J. Buttler

March 7, 2008

Encyclopedia of Database Systems

Disclaimer

Java Metadata Facility

David Buttler
Lawrence Livermore National Laboratory, http://people.llnl.gov/buttler1
This work (LLNL-JRNL-402071) was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Synonyms

Java Metadata Facility, JSR 175, Java Annotations

Definition

The Java Metadata Facility is introduced by Java Specification Request (JSR) 175 [1], and incorporated into the Java language specification [2] in version 1.5 of the language. The specification allows annotations on Java program elements: classes, interfaces, methods, and fields. Annotations give programmers a uniform way to add metadata to program elements that can be used by code checkers, code generators, or other compile-time or runtime components.

Annotations are defined by annotation types. These are defined the same way as interfaces, but with the symbol '@' preceding the 'interface' keyword. There are additional restrictions on defining annotation types:

1. They cannot be generic.
2. They cannot extend other annotation types or interfaces.
3. Methods cannot have any parameters.
4. Methods cannot have type parameters.
5. Methods cannot throw exceptions.
6. The return type of methods of an annotation type must be a primitive, a String, a Class, an annotation type, or an array, where the type of the array is restricted to one of the four allowed types.

See [2] for additional restrictions and syntax.

The methods of an annotation type define the elements that may be used to parameterize the annotation in code. Annotation types may have default values for any of its elements. For example, an annotation that specifies a defect report could initialize an element defining the defect outcome to 'submitted.' Annotations may also have zero elements. This could be used to indicate serializability for a class (as opposed to the current Serializability interface).

Main Text

There are several annotation types that are predefined in the Java 1.5 programming language: '@Override', '@Deprecated', and '@SuppressWarnings' are the most common ones.

'@Override' indicates that a method in a subclass overrides a method from its superclass, as opposed to overloading it. This is an example of an annotation with zero elements. A

common, yet difficult to identify, error in writing Java classes occurs when a programmer overloads the `equals` method, rather than overriding it. This leads to errors that are difficult to track down.

'@Deprecated' indicates that a class or method has been deprecated and that programmers should use an alternative. This replaces the javadoc '@deprecated' tag that served the same purpose.

'@SuppressWarnings' indicates that a compiler should not report warnings of a particular type. This particular annotation requires an element, such as '@SupressWarnings("unchecked")', defining the type of warning to ignore for the annotated compilation unit. Warning types are defined by the compiler and are not specified in the Java language specification.

Cross References
Metadata

References
1. Coward, D. (2004) JSR 175: A Metadata Facility for the Java™ Programming Language. 2004. http://jcp.org/en/jsr/detail?id=175

2. Gosling J., Joy B., Steele G., Bracha G. (2005) The Java™ Language Specification. Prentice Hall 2005.