Overview about the JSourceObjectizer

Wednesday, 02 April 2008
Last Updated Saturday, 09 January 2010

This document provides an overview about the JSourceObjectizer by the following sections:

-

What the JSourceObjectizer is and what it does

-

Features of the current JSourceObjectiser version and what's planned for future versions

-

An example of for what the JSourceObjectizer can be used

What the JSourceObjectizer is and what it does

In short, the JSourceObjectizer can be described as a Java parser wrapper. Actually the JSourceObjectizer consists of the following three different parts and only one of these parts contains a parser:

-

The JSOM (Java Source Object Model) interface types

This part can be seen as the JSourceObjectizer's frontend. Once a Java source has been parsed these interfaces provide all the methods to access the parser result and there's an interface type for most of the things that may occur within a Java source code. Examples for such interface types are MethodDefinition, LocalVariableDeclaration, IfStatement, ThrowsClause, ModifierList, Annotation, AnnotationElementValue, BinaryOperatorExpression and so on.

-

The JSourceObjectizer's unmarshaller

The unmarshaller part contains not only a real Java parser but also the implementations of the JSOM interface types. These implementations resolve the parser results and transform them into the appropriate JSOM types. The only type from the unmarshaller needed to be used by JSourceObjectizer users directly is the class Unmarshaller. All other classes within this part can be ignored for most cases - they just do their work.

-

The JSourceObjectizer's marshaller

Currently, this is just a placeholder for future versions of the JSourceObjectizer.

Back to the top

Features of the current JSourceObjectizer version and what's planned for future versions

JSourceObjectizer V1.xx

With the JSourceObjectizer V1.xx Java sources can be parsed and the parser results can be accessed easily. No

knowledge about grammars, parsers and so on is needed for that, i.e. it's just enough to be familiar with the syntax of the Java language.

JSourceObjectizer V2.xx

The next major version, JSourceObjectizer V2.xx, will also allow to modify the parser results and to write them back. Work on this new version has already been started but for some reasons the progress was very slow during the last year (the main reason was that paid jobs must be served first). However, due to the fact that some of the V2.xx features are needed urgently for a forthcomming project ... a paid job :-) ... working on the new version has a relatively high priority now.

Planned pre-releases

As long as there's no final release candidate of the new version pre-releases of early V2.xx development versions may be provided for downloading in the future whenever an internal minor milestone has been reached. Further information about this will be published as soon as a new timetable for the internal V2.xx minor milestones has been arranged.

Back to the top

An example of for what the JSourceObjectizer can be used

A very good example of for what the JSourceObjectizer can be used is a tool written for one of the major Munich software companies. This tool is based on the JSourceObjectizer V1.xx and its job is to check the compliance with internal programming guidelines from the semantical point of view.

Of course, a lot of such checks could also be done within a grammer definition, i.e. just in time when a Java source gets parsed and resolved, an example could be checking the condition 'System.exit() calls are not allowed' . However, semantical programming guidelines can have the nature to sound simple at first but become very complex if the compliance should be monitored automatically. This is the case for mostly all checks that need resolving type dependencies and/or managing symbol scope stacks. At a certain point of complexity it makes sense to have access to the parsed Java code with the help of a collection of interfaces that represent the content of a Java code in a more programmer-friendly way.

Back to the top