SableCC

von Yasushi Tomii

Einleitung

SableCC (www.sablecc.org) ist ein JAVA Compiler Generator. Mit Hilfe eines Grammar-File erzeugt er Lexer Klasse, Parser Klasse, Node Klasse und Analysis Klasse jeweils unter dem Verzeichnis mit eigenem Namen. Lexer- und Parserfunktion werden von einem Main- File realisiert. Möglicherweise läßt sich die Analysis Klasse darauf anwenden, AST zu bauen. Also muß man nur einen Grammar-File, einen Main-File und Hilfsfile(Falls man AST erzeugen möchte) schreiben.

Grammar-File

Der Grammar-File, durch denum diese vier Klassen erzeugt wurden, besteht aus 6 Teilen, nämlich *Package*, *Helper*, *States*, *Tokens*, *Ignored Token* und *Produktions*. Package zeigt den Pfad zu den Verzeichnissen für alle erzeugenden Klassen.

Lexer-Teil

Helper, *States*, *Tokens*, *Ignored Token* basiert auf DFA. Sie sind sogenannte Lexer-Teile Der Lexer-Teil arbeitet dafür Token tu erkennen. *Helper* ist eine Teilmenge der Token Definitionen. Nur die Token in dieser Menge sind anwendbar, um andere Token zu definieren. Im *Ignored Token*, der auch eine Teilmenge von Token ist, ist der beim Parsing zu ignorierende Token definiert.

Parser-Teil

Produktions ist der Hauptteil, der die LALR(1) Parser Klasse definiert. . Produktions paßt sich fast dem EBNF an d.h. *,+,? sind zwar erlaubt, aber Krämern ist nicht erlaubt z.B. P = (a |ab)* . Konflikte können dadurch vermieden werden, daß man direkt nach dem Gleichzeichen „=“ ein Kennzeichen in der geschleifte Klammern „{ ... }“ schreibt.

Main-File

Der Main-file muß die Lexer-, Parser- und Node-Klasse implementieren. Bei Erzeugung des ASTs muß er auch Hilfsfiles dafür implementieren.

Erzeugung der AST

Sablecc hat die Fähigkeit, mit Hilfe der Klasse Analysis einen AST zu erzeugen. Dazu muß einer (oder mehrere) zusätzliche Hilfsfile geschrieben werden. Sie muß Unterklasse von einem File in der Klasse Analysis sein. Meistens handelt es sich um eine Unterklasse des Files DeepFirstAdapter d.h. : Die Spezifikation eines ASTs kann flexibel definiert werden.

Bewertung

Die Regeln, nach denen man einen Grammatik-File schreibt, sind relativ simpel. Man braucht nur wenig zu tun, um Lexer- und Parserfunktion zu realisieren. Aber bei der Erzeugung des gewünschte AST werden Hilfsfile leicht größer. Das ist der Nachteil der Flexibilität der Erzeugung eines ASTs.