eqparser Reference

Willis L. Boughton

September 2026

---

eqparser is a package for parsing and evaluating math equations. A single equation or multiple dependent equations can be evaluated, and equations can include data from a database interface. The data can be scalar or one-dimension (1-D) array values, and scalar data can be saved back into the database interface. Multiple standard scalar and 1-D array functions are built in, e.g, square root and vector dot product, and custom functions can be added.

License

eqparser is distributed as freeware. It can be used on any number of computers by any number of users and may be distributed freely. eqparser is provided "as is", and any warranties of any kind, express or implied, for any use or intended use, on any computing system, are disclaimed, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose. In no event shall the software provider be liable for any damages of any form arising in any way out of the use of this software, even if advised of the possibility of such damage. By using eqparser, you acknowledge that you have read these terms, understand them, and agree to them.

Distribution

eqparser is distributed in Zip file eqparser-distrib.zip, which contains the following:

- JAR file eqparser.jar, a non-executable JAR containing all eqparser class files. eqparser.jar must be in the Java classpath.
- Package eqparser.demo, containing three files:

  1) Demo.java which has documented source code demonstrating many equation cases.

  2) DemoDb.java which demonstrates use of the eqparser database interface. (See the database interface section below.) Demo.java uses DemoDb.java.

  3) JAR file demo which executes Demo.java and DemoDb.java.

Getting Started

To get started with eqparser:

- Study this document.
- Study the eqparser.demo code, the javadoc, and run demo.jar.

eqparser Grammar

The `eqparser` grammar is:

```
formula = identifier '=' expression

identifier = letter{letter | digit | '_'}

letter = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' |
         'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' |
         'W' | 'X' | 'Y' | 'Z' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' |
         'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' |
         's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'

digit ='0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

expression = term | term '+' expression | term '-' expression

term = factor | factor '*' term | factor '/' term

factor = number | '-' factor | '(' expression ')' | dataid | function

number = integer | real

integer = ['+' | '-'] digit{digit}

real = integer '.' integer

function = identifier '(' function | arguments ')'

arguments = scalarArgs | twoIdArgs

scalarArgs = numberArgs | idArgs

numberArgs = number{',' number} (whether 1 or 2 args depends on function)

idArgs = dataid{',' dataid} (whether 1 or 2 args depends on function)

twoIdargs = dataid ',' dataid (for two-argument array functions only)
```

This grammar differs from some in that a `function` argument must consist only of numbers or identifiers, not a math operation (no `+`, `-`, `*`, or `/`). Array functions handle 1-dimensional arrays obtained from the database interface by identifier but do not handle an array of numbers directly in an equation. See the functions section later in this document.

The math operator hierarchy is the usual:

1) * (multiplication) and / (division)

2) + (addition) and – (subtraction)

Class Summary

Table 1 summarizes eqparser classes. Italics denotes an abstract class.

Table 1. Classes	
*Name**	*Description*
Equation	A parsed formula that can be evaluated. An Equation is created by the Parser; an Equation can not be instantiated directly by an application.
Equation.Function	A math function, as defined in the parser grammar. There are four types (see the javadoc API): ONE_SCALAR_ARG, TWO_SCALAR_ARGS, ONE_ARRAY_ARG, and TWO_ARRAY_ARGS. For each type there is a concrete subclass of Equation.Function.
Equation.Set	A set of equations to be evaluated simultaneously.
Evaluator	Evaluates an Equation or an Equation.Set, using a database interface to obtain data values specified in the equations.
Parser	Parses a math formula text string into an Equation. The parsing rules are the eqparser grammar.

Database Interface

By defining a database that implements interface Eqparser.Database (see the javadoc API), equations that contain data identifiers, not just numbers, can be evaluated. An example would be formula

```
"eq1 = 3 * scalar1 + vecdot(array1, array2)"
```

where scalar1 is a double value and array1 and array2 are double[]. By implementing an Eqparser.Database database, the Evaluator will retrieve these values from the database. The interface also enables storing a equation value into the database; this value could then be used in another Equation. The interface methods use an identifier to identify a data item; an example is interface method scalar:

```
public double scalar(String identifier) throws Excepton;
```

which retrieves a scalar (double) data item. The application must define what identifier means.

Classes eqparser.demo.Demo and eqparser.demo.DemoDb in the eqparser distribution demonstrate the use of the database interface. In the demo, identifier consists of two "sub" identifiers in the format groupName:itemName.