Engine expression language

https://documentation.dcr.design/documentation/engine-expression-language/

The Engine contains an expression language for computing guards and computational activities. The language is inspired by the FEEL language used in Decision Model and Notation, described by OMG.

The FEEL (Friendly Enough Expression Language) language makes it possible to convert data from one format to another, e.g. a string to a date, and write user friendly expressions. FEEL and DMN combined empowers the business user to write complex expressions.

Visit this example to learn how to compute birthday based on a Danish cpr number string.

Type: Null
Audience: Modelers, Programmers
Full Width Content:

Types

The languages supports the types Strings, Booleans, Integers, and Floats. `Null` is a valid value of every type.

Syntax

```
expr   ::= expr && expr         # Conjunction ("and")
         | expr || expr         # Disjunction ("or")
         | expr < expr
         | expr > expr
         | expr = expr
         | expr != expr
         | expr ++ expr         # String concatenation
         | expr + expr
         | ( expr )
         | const
         | var

 const ::= ('"'[^'"']*'"') | ('\''[^'\'']*'\'')        # string
         | ('-'? digit+)                               # int
         | ('-'? digit+ '.' digit+)                    # double
         | true
         | false
         | null

 var   ::= (char|['_' '$'])(char|['-' '_' '\'']|digit)* # identifier
```

Examples of constant expressions:

```
1
10.0
"A string"
null
```

```
true
false
```

Reserved keywords

The following list of reserved keywords cannot be used for activity ids.

Semantics

Variables `var` refer to events in scope at the location of the expression;
see [articles/location.md] for details.

Less-than and greater-than operators convert between numeric types, and return
false if either side is `null`.

Examples

An expression that evaluates to true if the event `Amount` is greater than 10000:

```
Amount > 10000
```

An expression that evaluates to true if the event `Type` is equal to the
string `Architect` or `null`.

```
Type == "Architect" || Type == null
```

An expression that evaluates to true if the event `Amount` plus 1000 is greater
than 10000 or less than 20000:

```
(Amount + 1000 > 10000) || (Amount + 1000 < 20000)
```

Programmatic specification

Expressions are constructed programmatically using the
Expression class.
Use the from-string constructor
which parses an expression from a string:

```
Expression e = new Expression("I > 1000");
```

Use expressions to construct guarded relations. E.g., to construct
a condition from top-level event “G” to top-level event “H” which
is active only when the top-level event “I” has a value greater than 1000:

```
graph.AddCondition("G", "H", null, new Expression("I > 0"), null);
```

You can check whether an expression is syntax and type-correct in a
particular graph using the call
CheckExpression.