表达式解析说明

# 词法规则

## 数据类型

支持五种数据类型：数字、日期、布尔、字符、字符串

### 数字

数字类型在解析过程中将被转化成BigDecimal类型，支持以下格式：

整数123；

小数123.333；

指数1.2E2 或 1.2e2（即120）

### 日期

支持两种形式的日期数据，一种是精确到日的，写法为[yyyy-MM-dd]，如[2012-03-31]；另一种精确到秒，写法为[yyyy-MM-dd HH:mm:ss]，如[2012-03-31 09:06:10]。

在表达式解析过程中日期类型将转化为java.util.Calendar。

### Boolean

布尔常量的写法可以是true、TRUE、false、FALSE。

### 字符

字符分普通字符和转义字符。

普通字符：’a’、’\_’

转义字符：’\n’、’\r’

### 字符串

字符串用双引号表示，可包含转义字符，如”Hello World”、”Hello \r\n World”。

## 变量

变量名以字母或下划线开头，只能包含字母、数字、下划线，如：abc、\_abc、\_a1。

## 操作符

算术操作符：+、 -、 \*、 /、 %（取余）；

逻辑操作符：&&、||、!

比较操作符：>、 >=、 <、 <=、 ==、 !=

赋值操作符：=

# 表达式使用

## 创建表达式

类Expression的实例表示一个表达式，可通过Expression的构造函数创建或者使用ExpressionFactory创建。

表达式由语句组成，每个语句必须由分号结尾。定义表达式时可包含一个语句或多个语句，解析结果是被执行的最后一个语句的结果。

Expression有三个构造函数，可分别接受参数Stirng、InputStream、Reader。下面以String为例创建表达式。

1. 使用构造函数创建

Expression exp = new Expression(“a=1; a + a\*100;”);

1. 使用ExpressionFactory创建

ExpressionFactory factory = ExpressionFactory.getInstance();

Expression exp = factory.getExpression("a=1; a + a\*100");

## 执行表达式

通过调用Expression的evaluate()方法执行表达式:

ExpressionFactory factory = ExpressionFactory.*getInstance*();

Expression exp = factory.getExpression("a=1;b=a\*100;");

Valuable result = exp.evaluate();

表达式的执行结果以接口Valuable表示，Valuable提供如下方法：

**public** DataType getDataType();

**public** **int** getIndex();

**public** BigDecimal getNumberValue();

**public** String getStringValue();

**public** Character getCharValue();

**public** Calendar getDateValue();

**public** Boolean getBooleanValue();

**public** Object getValue();

如果表达式的结果是数字，可以这样取值：

BigDecimal number = result.getNumberValue();

如果忽略执行结果的类型，可以通过getValue方法取得Object()的执行结果。通过getDataType方法可以获得执行结果的具体类型。

以下部分将详细介绍各种表达式的具体使用。

### 算术表达式

算术表达式只支持数字类型的常量、变量或表达式，可使用括号运算符：

100 / (2 \* 50);

100 / (50 + 50);

### 字符串连接

字符串和字符可以使用 + 连接。

例如，”Hello” + “World” + ‘!’;

### 逻辑表达式

逻辑表达式只支持布尔类型的常量、变量或表达式，可使用括号运算符：

( 2>1 || false) && (false || 1<2);

### 比较运算

支持数字、字符、字符串、日期类型的比较，不支持布尔类型的比较。

“zxc” > “abc”;

‘b’ >= ‘a’;

1+1 < 1;

[2012-03-31] > [2012-03-01];

### 赋值语句

第一次给变量赋值时确定了变量的类型，后续语句中不能给变量赋以不同的类型，否则抛出ArgumentsMismatchException异常。

例执行语句 a = “Hello Word”; a=1; 将会抛出异常：

Type mismatch in assignment: cannot convert from String to Number.

### 函数

#### 系统函数

表达式解析器目前只提供了三个系统函数：abs（取绝对值）、max（取最大值）和judge（判断）。

函数的调用支持嵌套：

Expression expression = factory.getExpression("1 + max(1,abs(-2)) + abs(-1);");

judge函数共接受三个参数，第一个参数为布尔类型，第二和第三个参数为任意类型，例如judge(2>1, “2”, “1”)将返回字符串”2”。

函数参数可以是固定个数的参数，如judge，也可以是任意多个参数，如abs和max

#### 自定义函数

通过继承抽象类Function可以自定义函数。

Function有两个构造函数Function(String) 和 function(String, DataType[])。Function(String)接受函数名，function(String, DataType[])接受函数名和参数类型数组。

继承Function还需要实现两个抽象方法：getArgumentNum()返回参数个数，execute(Valuable[])实现函数执行逻辑，并返回执行结果。

下面介绍如何自定义无参函数、固定参数函数和任意数量参数的函数。

1. 无参函数

定义无参函数时，getArgumentNum()返回0，且构造函数中无需传入参数类型数组。比如定义一个函数取得当前日期，函数名为getDate：

**public** **class** CurrentDate **extends** Function {

**public** CurrentDate() {

**super**("getDate");

}

**public** **int** getArgumentNum() {

**return** 0;

}

@Override

**protected** Object executeFunction(Valuable[] arguments)

**throws** ArgumentsMismatchException {

Calendar date = Calendar.*getInstance*();

date.setTime(**new** Date());

**return** date;

}

}

1. 固定参数函数

以上面提到的judge函数为例：

**public** **class** Judge **extends** Function {

**public** Judge() {

**super**("judge", **new** DataType[]{DataType.*BOOLEAN*, DataType.*ANY*, DataType.*ANY*});

}

**public** **int** getArgumentNum() {

**return** 3;

}

@Override

**protected** Object executeFunction(Valuable[] arguments)

**throws** ArgumentsMismatchException {

**boolean** condition = arguments[0].getBooleanValue();

**if**(condition)

**return** arguments[1].getValue();

**else**

**return** arguments[2].getValue();

}

}

getArgumentNum()返回3，构造函数中DataType.ANY表示参数为任意类型

1. 任意数量参数的函数

以上面提到的max函数为例：

**public** **class** Max **extends** Function {

**public** Max() {

**super**("max", **new** DataType[]{DataType.*NUMBER*});

}

**public** **int** getArgumentNum() {

**return** -1;

}

@Override

**public** Object executeFunction(Valuable[] arguments) **throws** ArgumentsMismatchException {

BigDecimal result;

**if**(arguments.length == 0) {

result = **new** BigDecimal("0");

} **else** {

result = arguments[0].getNumberValue();

**for**(**int** i=1; i<arguments.length; i++)

**if**(result.compareTo(arguments[i].getNumberValue()) < 0)

result = arguments[i].getNumberValue();

}

**return** result;

}

}

getArgumentNum()返回-1，表示参数为任意个数。对于任意数量参数的函数，参数必须为同一类型，所以在构造函数中传入的参数类型数组为：new DataType[]{DataType.NUMBER}。

#### 自定义函数注册

系统函数不需要注册，可以直接使用。使用自定义函数，必须先调用Expression的addFunction方法，将函数注册到表达式，才能使用。以上面的取当前日期函数为例，

Expression expression = factory.getExpression("getDate();");

expression.addFunction(new CurrentDate());

expression.evaluate();

### If语句

If语句的使用形式为：

if(Boolean) endif 或

if(Boolean) else endif

对于在分支内部定义的变量，其作用域仅为所在分支，在后续语句中不能直接使用，否则抛出变量为定义异常。

If语句的执行结果是最后一个被有效执行的语句，如：

if(2>1)

a=abs(-3);

if(false)

a=a+1;

else

a=a+2;

endif

a=a+1;

else

if(2>1)

a=9;

endif

endif

执行结果为6。

### 获得表达式中涉及的所有变量名

Expression提供函数getVariableNames()，以Set<String>返回表达式中的所有变量名。

### 获取变量值

表达式执行完成后，可以通过getVariableValue(String)取得变量的值，传入参数为变量名；还可以调用getVariableTable()取得所有变量的值，返回类型为Map<String, Valuable>，存储变量名和变量值的键值对。

### 设置变量值

表达式中某变量的初始值也可从外部设置，调用setVariableValue(String, Object);

Expression expression = factory.getExpression("a + 1;");

expression.setVariableValue("a", 1);

expression.evaluate(); //结果为2