Karith

Release 0.1.0

Franckyi

Aug 05, 2022

USAGE

Welcome to the Karith documentation website.

CHAPTER

ONE

INSTALLATION

1.1 Gradle (Groovy)

```
implementation 'dev.franckyi:karith:0.1.0'
```

1.2 Gradle (Kotlin)

```
implementation("dev.franckyi:karith:0.1.0")
```

1.3 Maven

```
<dependency>
    <groupId>dev.franckyi</groupId>
    <artifactId>karith</artifactId>
    <version>0.1.0</version>
</dependency>
```

CHAPTER

TWO

QUICKSTART

The Karith library is designed to be easy to use and highly configurable.

2.1 Simple example

```
val res = "1 + 2".result() // 3.0
val res = "3a + b".resultWith("a" to 1, "b" to 2) // 5.0
val res = "6 % 4".intResult() // 2
val res = "6 / min(x,y)".intResultWith("x" to 2, "y" to 3) // 3
```

> Note: The above example uses a global context. It is recommended to build and use your own context instead.

2.2 Creating a context

First of all, you need to create a context. A context is an object that contains all the operators, variables and functions that you can parse in an expression. A context can also cache expressions and their result.

A context is usually created by including some modules, operators, variables and functions in it. Modules are collections of operators, variables and functions. You can find more information about builtin modules, operators, variables and functions in the *Builtin elements* section.

You can easily create your own context using helper functions. Here is an example:

```
// creates a context with + and - operators only
val ctx = context { withOperators(Operators.PLUS, Operators.MINUS) }

// creates a context with basic operators (+, -, *, / and %)
val ctx = baseContext()

// creates a context with basic operators and functions
val ctx = defaultContext()
```

2.3 Parsing an expression

You can parse an expression using the `expression` or `expressionWith` method of a context.

```
val expr = ctx.expression("1 + 2")
val expr = ctx.expressionWith("3a + b", "a", "b")
val expr = ctx.expression("6 % 4")
val expr = ctx.expressionWith("6 / min(x,y)", "x", "y")
```

2.4 Computing a result

You can compute the result of an expression using the `result`, `intResult` `resultWith`, or `intResultWith` method of an expression.

```
val expr = ctx.expression("1 + 2")
val res = expr.result() // 3.0

val expr = ctx.expressionWith("3a + b", "a", "b")
val res = expr.resultWith("a" to 1, "b" to 2) // 5.0

val expr = ctx.expression("6 % 4")
val res = expr.intResult() // 2

val expr = ctx.expressionWith("6 / min(x,y)", "x", "y")
val res = expr.intResultWith("x" to 2, "y" to 3) // 3
```

CHAPTER

THREE

ADVANCED USAGE

TODO

3.1 Builtin elements

3.2 Cache

3.3 Internal algorithm

CHAPTER

FOUR

BUILDING

You can build Karith using these commands:

```
git clone https://github.com/Franckyi/Karith
cd Karith
./gradlew build
```

CHAPTER

FIVE

EXTERNAL LINKS

- KDoc
- GitHub repository
- Maven Central