Groovy リファレンス 概要

GROOVY-1.0

第 *4* 版 (2007 年 3 月)

このカードは、主に Groovy 言語プログラマーによって使用されることを目的とします。この内容に含まれる物は、オンラインドキュメント (http://groovy.codehaus.org) に含まれる言語基本情報から要約した内容になっています。カード内容は、時々しか更新されません。しかしながら、カードで引用された上記のマニュアルその他は、信頼できるリファレンスであり、最初に、変化を反映することになっています。Java から持ち越された指定を区別するために、Groovy での本質的に新しい指定名前は、イタリックで示されます。

この内容についてのコメントは、下記のアドレスにお送りください。

このかなり微妙な翻訳内容についてのご意見は、T.Yamamoto へ。

Groovy Technical Publications Systems

groovy.codehaus.org

キーワード

文法

as	import *type* as *id*
assert	assert *expr expr?*
break	break *lbl?*
case	switch *expr* case *expr stmt**
catch	try *stmt** catch *type id stmt**
class	*mod** class *id*
continue	continue *lbl?*
def	def *methodDeclaration*
default	switch *expr* case default*stmt**
do	do *stmt** while *expr*
else	if *expr stmt** else if *expr stmt**
extends	*mod** class *id* extends *type*
finally	try *stmt** finally *stmt**
for	for *expr**;*expr*;*expr stmt**
for	for *id* in *id stmt**
if	if *expr stmt** else if *expr stmt**
in	for *id* in *id stmt**
implements	*mod** class *id* implements *type**
import	import *type*
instanceof	*expr* instanceof *type*
interface	*mod** interface *id*
new	new *type*
package	package *id*
property	*mod** property *type? id*
return	return *expr?/*
switch	switch *expr* case *expr stmt**
throw	throw *expr*
throws	*methodDeclaration* throws *type*
try	try *stmt** catch *type id stmt**
while	do *stmt** while *expr*
	while *expr stmt**

修飾子

abstract	abstract class *id*
Annotations	@Annotation
Annotations	@Annotation(x = 3, y = 5)

リストとマップ

type[] *id* = new *type*[*size*]

演算子

id = *expr*	**代入**
expr & *expr*	**ビット積**
id &= *expr*	**ビット積 代入**
id = ∼ *expr*	**ビット否定**
expr | *expr*	**ビット和**
id |= *expr*	**ビット和 代入**
id = *expr* >>> *expr*	**ビット右シフト**
id >>>= *expr*	**ビット右シフト 代入**

Groovy JDK

コレクションとプロパティー Collections and properties

注: 文章内の *cltn* は、リスト, セット,*matcher*, 文字列,*char* シーケンス, 配列も含まれます.

cltn [index|indices|range|property]
指定した (場所 | 位置 | 領域) のオブジェクトを得る.
obj [index|property] = value
指定した場所に *value* を代入
cltn << obj
コレクションに *obj* を追加
cltn + obj
コレクションに *obj* を追加
list − cltn
リストからアイテムを除去
cltn * num
コレクション内のアイテムを回数分繰り返す

obj. **getProperties**()
obj から Map のプロパティを得る
obj. **getMetaPropertyValues**()
obj からメタプロパティリストを得る
cltn. **count**(obj) *
コレクション内に存在する *obj* の数を数える
map. **get**(key, defaultValue) *
cltn. **size**() *

cltn. **collect**() {closure} *
closure を変換して新しいコレクションを作成
obj. **each**() {closure} *
closure の内容でオブジェクトの繰り返し処理
obj. **eachWithIndex**() {closure}
closure の内容を番号付きでオブジェクトの繰り返し処理
obj. **find**() {closure} *
closure の条件に一致した最初のアイテムを探す
obj. **findAll**() {closure} *
closure の条件に一致した全てのアイテムを返す
obj. **findIndexOf**() {closure}
return first index that matches condition *closure*
obj. **grep**(regex|range|etc..) *
cltn. **inject**(value) {closure}
returns closure(closure(closure(value,item0) ,item1) ,item2) ...
cltn. **max**([comparator]) {closure} *
コレクション内の最大値を返す
cltn. **min**([comparator]) {closure} *
コレクション内の最小値を返す
list. **reverseEach**() {closure}
closure の内容でオブジェクトの繰り返し処理を逆順で行う
cltn. **sort**([comparator]) *
コレクションをリストへ並べ替える, オプションで *comparator* も使用可能
cltn. **sort**() {closure} *
クロージャを比較条件にしてコレクションをリストへ並べ替える

cltn. **asImmutable**()
cltn. **asSynchronized**()
list. **flatten**()
list. **intersect**(cltn)
リストとコレクションの共通内容を返す
cltn. **join**(separator) *
cltn の全てのエレメントを連結して文字列にする
list. **pop**() *
リストから最後のアイテムを削除して返す
cltn. **reverse**()
map. **subMap**(keys)
与えられたキーのマップを返す
cltn. **toList**()

文字列 Strings

str ++
str −−
str + obj
str と *obj* 連結する
str − obj
最初の *obj* を *str* から除去
str << value
str. **padRight**(size,[padding])
size の範囲で左揃え
str. **center**(size, [padding])
size の範囲で中央揃え
str. **padLeft**(size,[padding])
size の範囲で右揃え
str. **contains**(str2) *
str が *str2* を含む場合 true を返す
str. **eachMatch**(regex) {closure} *
指定された正規表現 *regex* に一致した内容に *closure* を適用
str. **toCharacter**()
str. **toList**()
str. **toLong**()
str. **toURL**()
str. **tokenize**([token]) *

入出力 Input/output

注: ここでの *url* 意味は、*url, file, stream,reader* も含まれます.

dir. **eachFile**() {closure}
dir 内のそれぞれのファイルに *closure* 内容を適用
dir. **eachFileRecurse**() {closure}
再帰的に, *dir* 内のそれぞれのファイルに *closure* 内容を適用
url. **eachByte**() {closure}
file の各バイトに *closure* 内容を適用
url. **eachLine**() {closure}
入力各行に *closure* 内容を適用
file. **readBytes**()
in. **readLine**()
in から単一行, 全行の読込
file. **readLines**()
file から行の List を取得
url. **getText**([charset])
リソースから全てのテキストを取得
file. **splitEachLine**(regEx) {closure}
regEx で分割された *file* の各行に *closure* を適用
url. **withReader**() {closure}
closure を *in* に適用して *in* を閉じる
out << obj
obj を stream, process また socket に付加
file. **append**(text, [charset])
bytes. **encodeBase64**()
str. **decodeBase64**()
in. **filterLine**([out]) {closure}
in から読み込み、*closure* に一致する行のみ *out* へ書き出す
in. **transformChar**(out) {closure}
in. **transformLine**(out) {closure}
in から読み込み、*out* 書きだし時に各行に *closure* を適用
file. **withOutputStream**() {closure}
file. **withPrintWriter**() {closure}
out. **withStream**() {closure}
skt. **withStreams**() {closure}
out. **withWriter**([charset]) {closure}
file. **withWriterAppend**(charset) {closure}
file. **write**(text, [charset]) *
out. **writeLine**(line)

その他

date ++
date −−
date + days
date − days
obj. **dump**() *
obj の詳細ダンプ文字列を返す
obj. **inspect**() *
obj のインスタンス作成時に使用された groovy expression を返す
obj. **invokeMethod**(method, args) *
obj. **print**(obj) *
obj. **print**(out) *
obj. **println**(object) *
obj. **println**(out) *
num. **step**(endNum, stepNum) {closure}
closure 内容を num で開始して *endNum* まで繰り返す
num. **times**() {closure} *
closure 内容を num 回数繰り返す
num. **upto**(endNum) {closure}
closure 内容を num から *endNum* まで繰り返す
obj. **use**(categoryClass) {closure} *
指定したクラスに *closure* を結びつける
obj. **use**(categoryClassList) {closure} *
指定した複数クラスに *closure* を結びつける
ps. **waitForOrKill**(milliSecs)
ps. **getText**()
Thd. ***start***() {closure}
Thd. ***startDaemon***() {closure}
Mtr. ***getLastMatcher***()

Groovy Developers Kit

Groovy SQL

Sql(datasource|connection|sql)
newInstance(url,[user],[pass],[driver])
sql. **call**(str,[params])
sql. **eachRow**(str,[params]) {closure}
sql. **execute**(str,[params])
sql. **executeUpdate**(str,[params])
sql. **close**()

スニペット Snippets

Builder Markup, Node, StreamingMarkup, DOM, Ant, JavaDoc, Swt, JFace, Swing

```
farm = new NodeBuilder().farm(){animal(type:'pig')}
```

GPath walks the tree

```
farm.animal.collect{it['@type']}.contains('pig')
```

Groovy コード例

```
package com.example
import org.example.Cow
// これは、Groovy のサンプルです。
class HighlandCow extends Cow {
  void mooAtPeople() {
    sql = Sql.newInstance("jdbc:foo:bar")
    sql.eachRow("select * from PERSON") {
      println("Och-Aye ${it.firstname}")
    }
  }
}
```

ツール

Ant タスク

```
<taskdef name="groovyc"
         classname="org.codehaus.groovy.ant.Groovyc"
         classpathref="my.classpath"/>
<groovyc destdir="${build.classes.dir}"
         srcdir="${src.dir}"  listfiles="true">
  <classpath refid="my.classpath"/>
</groovyc>
```

コマンドラインツール

> **groovy** [options] cheese.groovy [args]
指定された groovy スクリプトを解析して実行
> **groovyc** [options] cheese.groovy
指定された groovy スクリプトをコンパイル
> **groovysh**
対話式の groovy セッションを開始
> **groovyConsole**
GUI ベースの groovy セッションを開始

$Revision: 5826 $, $Date: 2007-03-30 22:21:17 +0000 (金, 30 3 2007) $.
http://javanicus.com/