



Groovy Reference Summary

GROOVY-1.0-BETA-7

Second Edition (September 2004)

This card is intended primarily for use by Groovy language programmers. It contains basic language information summarized from the online documentation (<http://groovy.codehaus.org>). The card will be updated from time to time. However the above manual and others cited on the card are the authoritative reference sources and will be first to reflect changes.

To distinguish them from instructions carried over from the Java Language, the names of instructions essentially new with Groovy are shown in italics.

Comments about this publication may be sent to the address below.

Groovy Technical Publications Systems

groovy.codehaus.org

Keywords

Grammar

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

Groovy JDK

Collections and properties

Note: cltn in this sense can include lists, sets, matchers, strings, charSeqs and arrays

```
cltn [index|indices|range|property ]
    obtains objects at specified location
obj [index|property] = value
    put value at location
cltn << obj
    append obj to collection
cltn + obj
list - cltn
cltn * num
    repeat items in collection a number of times
```

```
obj. allProperties()
    obtain List of properties on obj
cltn. count(obj) *
    counts number of occurrences of obj in collection
map. get(key, defaultValue) *
cltn. size() *
```

```
cltn. collect() {closure} *
    new collection of closure transformed items
obj. each() {closure} *
    iterate through object applying closure
obj. eachProperty() {closure}
    apply closure to each property of obj
obj. eachPropertyName() {closure}
obj. eachWithIndex() {closure}
    iterate through object with a counter applying closure
obj. find() {closure} *
    find first item picked by closure condition
obj. findAll() {closure} *
    returns all items picked by closure condition
obj. findIndexOf() {closure}
    return first index that matches condition closure
obj. grep(regex|range|etc..) *
cltn. inject(value) {closure}
    returns closure( closure(value,item0),item1),item2) ...
cltn. max([comparator]) {closure} *
    returns the maximum value found in the collection
cltn. min([comparator]) {closure} *
    returns the minimum value found in the collection
list. reverseEach() {closure}
    iterate backwards through list applying closure
cltn. sort([comparator]) *
    sorts collection into a list, optionally using a comparator
cltn. sort() {closure} *
    sorts collection into a list using closure as comparator
```

```
cltn. asImmutable()
cltn. asSynchronized()
list. flatten()
list. intersect(cltn)
    returns intersection of list and collection
cltn. join(separator) *
    concatenate all the elements of cltn into a string
list. pop() *
    remove and return last item from list
cltn. reverse()
map. subMap(keys)
    returns a map of the given keys
cltn. toList()
```

Strings

```
str  ++
    increment the number at end of string
str  --
    reduce the number at end of string
str  + obj
str  - obj
    remove the first obj from str
str  << value
str.  padRight(size,[padding])
    left justifies string padded out to size
str.  center(size, [padding])
    centers a string padded out to size
str.  padLeft(size,[padding])
    right justifies string padded out to size
str.  contains(str2) *
    true if str contains str2
str.  eachMatch(regex) {closure} *
    apply closure to each match of the specified regex
str.  toCharacter()
str.  toList()
str.  toLong()
str.  toURL()
str.  tokenize([token]) *
```

Input/output

Note: url in this sense can include urls, files, streams and readers

```
dir.  eachFile() {closure}
    apply closure to each file in dir
dir.  eachFileRecurse() {closure}
    apply closure to each file in dir recursively

url.  eachByte() {closure}
url.  eachLine() {closure}
    apply closure to each line of input
file.  readBytes()
in.    readLine()
    read a single, whole line from in
file.  readLines()
    obtain List of lines from file
url.  getText([charset])
    fetch all available text from resource
file.  splitEachLine(regex) {closure}
    read in each line of file, apply closure to data delimited by regex
url.  withReader() {closure}
    apply closure to in, then close in
```

```
out  << obj
    append obj to stream, process or socket
file.  append(text, [charset])
bytes.encodeBase64()
str.   decodeBase64()
```

```
in.   filterLine([out]) {closure}
    read from in and write each line to out only if closure
in.   transformChar(out) {closure}
in.   transformLine(out) {closure}
    read in and apply closure to each line being written out
file.  withOutputStream() {closure}
file.  withPrintWriter() {closure}
out.   withStream() {closure}
skt.   withStreams() {closure}
out.   withWriter([charset]) {closure}
file.  withWriterAppend(charset) {closure}
    apply closure to a new appending Writer on file, then close
file.  write(text, [charset]) *
out.   writeLine(line)
```

Misc

```
date  ++
date  --
date  + days
date  - days
obj.   dump() *
    returns a detailed dump string of obj
obj.   inspect() *
    returns the groovy expression used to create this instance of obj
obj.   invokeMethod(method, args) *
obj.   print(obj) *
obj.   print(out) *
obj.   println(object) *
obj.   println(out) *
num.   step(endNum, stepNum) {closure}
    iterates closure starting at this number, stepping up to endNum
num.   times() {closure} *
    iterates closure num times
num.   upto(endNum) {closure}
    iterates closure starting at this number, up to endNum
obj.   use(categoryClass) {closure} *
    attach closure to specified class
obj.   use(categoryClassList) {closure} *
    attach closure to specified classes
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()
```

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')
```

Sample Groovy code

```
package com.example

import org.example.Cow

/**
 * This is a sample of Groovy
 */
class HighlandCow extends Cow {

    void mooAtPeople() {
        sql = Sql.newInstance("jdbc:foo:bar")
        sql.eachRow("select * from PERSON") {
            println("Och-Aye ${it.firstname}")
        }
    }
}
```

Tools

Ant task

```
<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>
```

Command Line Tools

```
> groovy [options] cheese.groovy [args]
    interpret and execute specified groovy script
> groovyc [options] cheese.groovy
    compiles specified groovy script
> groovysh
    begins an interactive groovy session
> groovyConsole
    begins a GUI based groovy session
```