内容目录

Groovy 学习笔记

一、 多线程的运用

```
t = new Thread() { /* Closure body */ }
t.start()

Thread.start { /* Closure body */ }

Thread.startDaemon { /* Closure body */ }

new Timer().runAfter(1000){ /* Closure body */}
```

Listing 9.11 Using threads with synchronization for the producer/consumer problem

```
class Storage {
    List stack = []
    synchronized void leftShift(value){
        stack << value
        println "push: $value"
        notifyAll()
    }
    synchronized Object pop() {
        while (stack.isEmpty()) {
            try{ wait() }
            catch(InterruptedException e){}
        }
        def value = stack.pop()
        println "pop : $value"
        return value
    }
}
storage = new Storage()
Thread.start {
    for (i in 0..9) {
        storage << i
        sleep 100
    }
}

Thread.start {
    10.times {
        sleep 200
        value = storage.pop()
    }
}
```

二、 处理外部进程

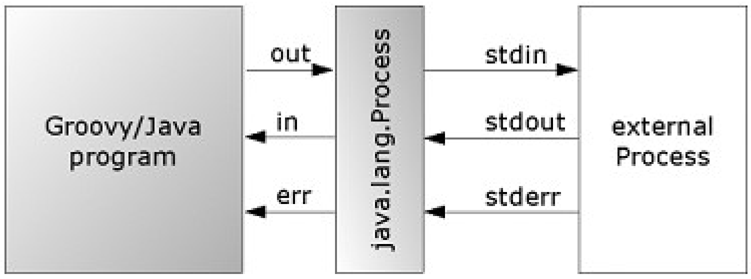
```
Process proc = myCommandString.execute()
```

例 1

```
def dircmd = ['cmd','/c','dir']//定义命令

def dir    = /\Program Files/ //定▪ ▪行目▪

def proc   = (dircmd + dir).execute() //▪行命令
```

例 2： window 下设置环境变量

```
def env = ['USERNAME=mittie']

def proc = 'cmd /c set'.execute(env, new File('/'))

println proc.text //打印输出文本

//种定向输出流

InputStream  in  = proc.in

InputStream  err = proc.err

OutputStream out = proc.out
```



三、 模板使用

Listing 9.14 Using a simple template engine for email text

```
mailReminder = '''
Dear ${salutation?salutation+' ':''}$lastname,
another month has passed and it's time for these
<%=tasks.size()%> tasks:
<% tasks.each { %>- $it
<% } %>
your collaboration is very much appreciated
'''
def engine   = new groovy.text.SimpleTemplateEngine()
def template = engine.createTemplate(mailReminder)
def binding  = [
  salutation: 'Mrs.',
  lastname  : 'Davis',
  tasks     : ['visit the Groovy in Action (GinA) page',
         'chat with GinA readers']
]
assert template.make(binding).toString() == '''
Dear Mrs. Davis,
another month has passed and it's time for these
2 tasks:
- visit the Groovy in Action (GinA) page
- chat with GinA readers
```

```
your collaboration is very much appreciated
'''
```

四、 使用 Groovlets

Listing 9.15 Sample web.xml file for configuring a web application for Groovlet use

```
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd" >
<web-app>
  <display-name>Groovlet Demonstrator</display-name>
  <description>
    Showing the use of Groovlets for Groovy in Action
  </description>
  <servlet>
    <servlet-name>Groovy</servlet-name>
    <servlet-class>groovy.servlet.GroovyServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Groovy</servlet-name>
    <url-pattern>*</url-pattern>
  </servlet-mapping>
</web-app>
```

Table 9.6 Information available to Groovlets

Name	Note	Example usage
headers	Map of HTTP request headers	headers.host
params	Map of HTTP request parameters	params.myParam
session	ServletSession, can be null	session?.myParam
request	HttpServletRequest	request.remoteHost
response	HttpServletResponse	response.contentType='text/xml'
context	ServletContext	context.myParam
application	ServletContext (same as context)	application.myParam
out	response.writer	Lazy init, not in binding
sout	response.outputStream	Lazy init, not in binding
html	Builder initialized as new MarkupBuilder(out)	Lazy init, not in binding

session 的使用

```
if (!session) // error handling here if needed
session = request.session
```

Listing 9.18 Groovlet code of the HighLow game

```
def session = request.session
def guess   = params.guess
guess = guess ? guess.toInteger() : null
if (params.restart) guess = null
if (!session.goal || params.restart) {
  session.goal = (Math.random()*100).toInteger()
}
def goal = session.goal
html.html{ head { title 'Think of a Number'  }
  body {
    h1 'Think of a Number'
    if (goal && guess) {
      div "Your guess $guess is "
      switch (guess) {
        case goal        : div 'correct!'; break
        case {it < goal} : div 'too low' ; break
        case {it > goal} : div 'too high'; break
      }
    }
    p "What's your guess (0..100)?"
    form(action:'NumberGuesser.groovy'){
      input(type:'text', name:'guess', '')
      button(type:'submit', 'Guess')
      button(type:'submit', name:'restart', value:'true',
          'New Game')
}   }   }
```

使用模板生成▪面 Templating Groovlets

web.xml

```
<servlet>
  <servlet-name>template</servlet-name>
  <servlet-class>groovy.servlet.TemplateServlet</servlet-
class>
</servlet>
<servlet-mapping>
  <servlet-name>template</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

Listing 9.19 Number.template.html as a view for the HighLow game

```
<html>
```

```
<head>
 <title>Think of a Number</title>
</head>
<body>
 <h1>Think of a Number</h1>
 Your guess $guess is <%
   switch (guess) {
     case goal         : out << 'correct!'; break
     case {it < goal} : out << 'too low' ; break
     case {it > goal} : out << 'too high'; break
   }
 %>
 <p>What&quot;s your guess (0..100)?</p>
 <form action='Templater.groovy'>
  <input type='text' name='guess'>
  <button type='submit'>Guess</button>
  <button type='submit' name='restart' value='true'>New Game
  </button>
 </form>
</body>
</html>
```

groovy:

```
def engine   = new groovy.text.SimpleTemplateEngine()
def source   = getClass().classLoader.
               getResource('/Number.template.html')
def template = engine.createTemplate(source)
out << template.make(goal:50, guess:49)
```

五、 数据库编程

Table 10.1 HSQLDB subprotocols

URL pattern	Purpose
`jdbc:hsqldb:hsql://server/dbname`	Connects to a HSQLDB server process; use when multiple clients or processes need to share the database
`jdbc:hsqldb:file:/dir/dbname`	Connects to a single-client HSQLDB instance with file-based persistence; multiple files starting with *dbname* will be created if the database doesn't yet exist
`jdbc:hsqldb:mem:dbname`	Connects to a nonpersistent in-memory database

- ◆ 初始化连接

```
import groovy.sql.Sql
db = Sql.newInstance(
  'jdbc:hsqldb:mem:GinA',
  'sa',
  '',
  'org.hsqldb.jdbcDriver')
```

```
source = new org.hsqldb.jdbc.jdbcDataSource()
source.database = 'jdbc:hsqldb:mem:GinA'
source.user     = 'sa'
source.password = ''
db = new groovy.sql.Sql(source)
```

- ◆ 执行 sql

```
db.execute(statement)
db.execute '''
  CREATE TABLE Athlete (
    firstname   VARCHAR(64),
    lastname    VARCHAR(64),
    dateOfBirth DATE
  )
'''
```

```
db.execute '''
  CREATE TABLE Athlete (
    athleteId   INTEGER GENERATED BY DEFAULT AS IDENTITY,
    firstname   VARCHAR(64),
    lastname    VARCHAR(64),
    dateOfBirth DATE
  );
  CREATE INDEX athleteIdx ON Athlete (athleteId);
'''
db.execute '''
  DROP   INDEX athleteIdx IF EXISTS;
  DROP   TABLE Athlete    IF EXISTS;
  CREATE TABLE Athlete (
    athleteId   INTEGER GENERATED BY DEFAULT AS IDENTITY,
    firstname   VARCHAR(64),
    lastname    VARCHAR(64),
    dateOfBirth DATE
  );
  CREATE INDEX athleteIdx ON Athlete (athleteId);
'''
db.execute '''
  INSERT INTO Athlete (firstname, lastname,     dateOfBirth)
        VALUES ('Paul',    'Tergat',     '1969-06-17');
  INSERT INTO Athlete (firstname, lastname,     dateOfBirth)
        VALUES ('Khalid',  'Khannouchi', '1971-12-22');
  INSERT INTO Athlete (firstname, lastname,     dateOfBirth)
        VALUES ('Ronaldo', 'da Costa',   '1970-06-07');
'''
```

▪ **参数**

```
String athleteInsert = '''
```

```
  INSERT INTO Athlete (firstname, lastname, dateOfBirth)
        VALUES  (?, ?, ?);
'''
db.execute athleteInsert, ['Paul',    'Tergat',     '1969-06-
17']
db.execute athleteInsert, ['Khalid',  'Khannouchi', '1971-12-
22']
db.execute athleteInsert, ['Ronaldo', 'da Costa',   '1970-06-
07']

def athletes = [
  [first: 'Paul',    last: 'Tergat',     birth: '1969-06-
17'],
  [first: 'Khalid',  last: 'Khannouchi', birth: '1971-12-
22'],
  [first: 'Ronaldo', last: 'da Costa',   birth: '1970-06-07']
]
athletes.each { athlete ->
 db.execute """
  INSERT INTO Athlete (firstname, lastname, dateOfBirth)
   VALUES (${athlete.first}, ${athlete.last}, $
{athlete.birth});
 """
}
```

◆ 设置 logger

```
import java.util.logging.*
Logger.getLogger('groovy.sql').level = Level.FINE
// your db.execute(GString)Basic database operations 333
This produces
30.10.2005 19:08:27 groovy.sql.Sql execute
```

```
FINE:
  INSERT INTO Athlete (firstname, lastname, dateOfBirth)
         VALUES (?, ?, ?);
```

table 10.2 Versions of the execute method

Returns	Method name	Parameters
boolean	execute	String statement
boolean	execute	String prepStmt, List values
boolean	execute	GString prepStmt
int	executeUpdate	String statement
int	executeUpdate	String prepStmt, List values
int	executeUpdate	GString prepStmt

Table 10.3 Methods for reading data from the database

Returns	Method	Parameters
void	eachRow	String statement { row -> code }
void	eachRow	String prepStmt, List values { row -> code }
void	eachRow	GString prepStmt { row -> code }

Table 10.3 Methods for reading data from the database *(continued)*

Returns	Method	Parameters
void	query	String statement { resultSet -> code }
void	query	String prepStmt, List values { resultSet -> code }
void	query	GString prepStmt { resultSet -> code }
List	rows	String statement
List	rows	String prepStmt, List values
Object	firstRow	String statement
Object	firstRow	String prepStmt, List values

```
db.eachRow('SELECT * FROM Athlete'){ athlete ->
  println athlete.firstname + ' ' + athlete.lastname
  println 'born on '+ fmt.format(athlete.dateOfBirth)
  println '-' * 25
}
```

```
db.eachRow('SELECT firstname, lastname FROM Athlete'){ row ->
  println row[0] + ' ' + row[1]
}
db.query('SELECT firstname, lastname FROM Athlete'){ resultSet
->
  if(resultSet.next()){
    print   resultSet.getString(1)
    print   ' '
    println resultSet.getString('lastname')
  }
}
List athletes = db.rows('SELECT firstname, lastname FROM
Athlete')
println "There are ${athletes.size()} Athletes:"
println athletes.collect{"${it[0]} ${it.lastname}"}.join(", ")



Listing 10.1  CRUD operations with Groovy

dbHandle = null
def getDb() {
  if (dbHandle) return dbHandle
  def source = new org.hsqldb.jdbc.jdbcDataSource()
  source.database = 'jdbc:hsqldb:mem:GIA'
  source.user = 'sa'
  source.password = ''
  dbHandle = new Sql(source)
  return dbHandle
}
def reset() {
  db.execute '''
    DROP   INDEX athleteIdx IF EXISTS;
    DROP   TABLE Athlete IF EXISTS;
    CREATE TABLE Athlete (
      athleteId   INTEGER GENERATED BY DEFAULT AS IDENTITY,
      firstname   VARCHAR(64),
      lastname    VARCHAR(64),
      dateOfBirth DATE
    );
    CREATE INDEX athleteIdx ON Athlete (athleteId);
  '''
}
def create(firstname, lastname, dateOfBirth) {
  db.execute """
    INSERT INTO Athlete ( firstname, lastname, dateOfBirth)
          VALUES ($firstname,$lastname,$dateOfBirth);
  """
}
def findAll() {
  db.rows 'SELECT * FROM Athlete'
}
def updateFirstName(wrong, right) {
  db.execute """
    UPDATE Athlete
      SET firstname = $right WHERE firstname = $wrong;
  """
}
def delete(firstname) {
  db.execute "DELETE FROM Athlete WHERE firstname =
$firstname;"
}
reset()
assert ! findAll(), 'we are initially empty'
```

```
create 'Dirk', 'Koenig', '1968-04-19'
assert 'Dirk'  == findAll()[0].firstname
updateFirstName  'Dirk', 'Dierk'
assert 'Dierk' == findAll()[0].firstname
delete 'Dierk'
assert ! findAll(), 'after delete, we are empty again'
```

◆ 使用 DataSet

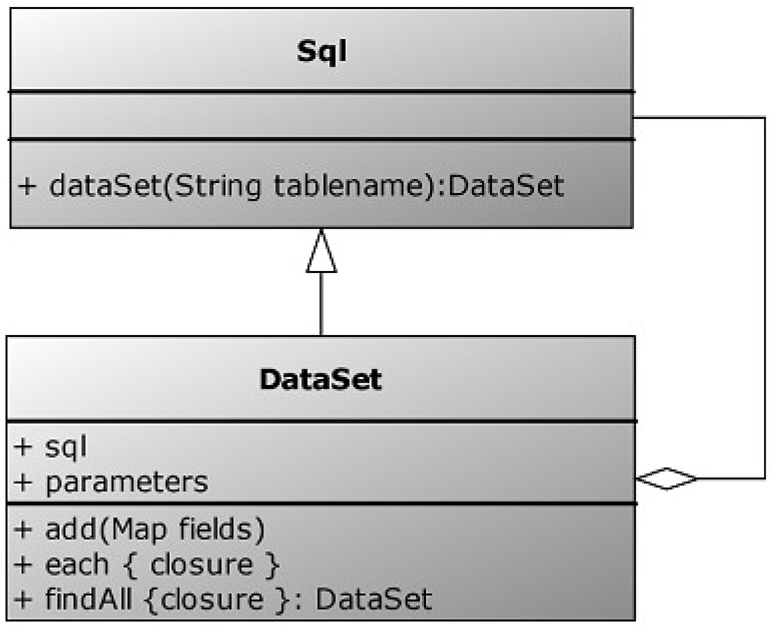

Sql
+ dataSet(String tablename):DataSet
DataSet
+ sql
+ parameters
+ add(Map fields)
+ each { closure }
+ findAll {closure }: DataSet

Figure 10.2 UML class diagram of groovy.sql.DataSet decorating groovy.sql.Sql

```
// if db refers to an instance of Sql
athleteSet = db.dataSet('Athlete')
athleteSet.add(
  firstname:   'Paula',
  lastname:    'Radcliffe',
  dateOfBirth: '1973-12-17')
athleteSet.each {
  println it.firstname
}
```

```
athleteSet.findAll{ it.dateOfBirth > '1970-1-1'
youngsters = athleteSet.findAll{ it.dateOfBirth > '1970-1-1' }
youngsters.each { println it.firstname }

youngsters = athleteSet.findAll{ it.dateOfBirth > '1970-1-1' }
println youngsters.sql
println youngsters.parameters
youngsters.each { println it.firstname }

youngsters = athleteSet.findAll{ it.dateOfBirth > '1970-1-1' }
paula      = youngsters.findAll{ it.firstname == 'Paula' }
println paula.sql
println paula.parameters

youngsters = athleteSet.findAll{
  it.dateOfBirth > '1970-1-1' && it.firstname == 'Paula'
}
```

Table 10.4 Mapping of Groovy AST nodes to their SQL equivalents

AST node	SQL equivalent
`&&`	`and`
`||`	`or`
`==`	`=`
Other operators	Themselves, literally
`it.`*`propertyname`*	`propertyname`
Constant expression	`?` (Expression is added to the parameters list)

Listing 10.2 Athlete example infrastructure: DbHelper

```
import groovy.sql.Sql
import groovy.text.SimpleTemplateEngine as STE
```

```
class DbHelper {
  Sql db
  DbHelper() {
    def source = new org.hsqldb.jdbc.jdbcDataSource()
     source.database = 'jdbc:hsqldb:mem:GIA'
     source.user = 'sa'
     source.password = ''
     db = new Sql(source)
  }
  def simpleTemplate = new STE().createTemplate('''
   DROP    INDEX ${lowname}Idx IF EXISTS;
   DROP    TABLE $name     IF EXISTS;
   CREATE TABLE $name (
      ${lowname}Id     INTEGER GENERATED BY DEFAULT AS
    IDENTITY, $fields
   );
   CREATE INDEX ${lowname}Idx ON $name (${lowname}Id);''')
  def executeDdl(DataAccessObject dao) {
       def template = simpleTemplate
       def binding = [
           name:     dao.tablename,
           lowname: dao.tablename.toLowerCase(),
           fields:  dao.schema.collect{ key, val ->
             "     ${key.padRight(12)} $val" }.join(",\n")
         ]
       def stmt = template.make(binding).toString()
       db.execute stmt
     }
   }
```

Listing 10.3 Athlete example infrastructure: DataAccessObject

```
abstract class DataAccessObject {
  Sql db
  abstract List getFields()
  def dataSet()     { db.dataSet(tablename) }
  def getIdField() { tablename.toLowerCase() + 'Id' }
  def getWhereId() { "WHERE $idField = ?"}
  String getTablename() {
    def name = this.getClass().name
    return name[name.lastIndexOf('.')+1..-4]
  }
  def create(List args) {
    Map argMap = [:]
    args.eachWithIndex { arg, i -> argMap[fieldNames[i]] =
arg }
    dataSet().add argMap
  }
  Map getSchema() {
    Map result = [:]
    fieldNames.each {result[it] = fields[fields.indexOf(it)
+1]}
    return result
  }
  List getFieldNames() {
    List result = []
    0.step(fields.size(),2) { result << fields[it] }
    return result
  }
  def update(field, newValue, id) {
    def stmt = "UPDATE $tablename SET $field = ? $whereId"
    db.executeUpdate stmt, [newValue, id]
  }
  def delete(id) {
    def stmt = "DELETE FROM $tablename $whereId"
    db.executeUpdate stmt, [id]
  }
  def all(sortField) {
    def selects = fieldNames + idField
    def result = []
    def stmt = "SELECT " + selects.join(',') +
      " FROM $tablename ORDER BY $sortField"
    db.eachRow(stmt.toString()){ rs ->
      Map businessObject = [:]
      selects.each { businessObject[it] = rs[it] }
      result << businessObject
    }
    return result
  }
}
```

Listing 10.4 Athlete example infrastructure: AthleteDAO

```
class AthleteDAO extends DataAccessObject {
  List getFields() {
    return [
      'firstname',    'VARCHAR(64)',
      'lastname',     'VARCHAR(64)',
      'dateOfBirth',  'DATE'
    ]
  }
}

domain model

class Athlete {
  def firstname
  def lastname
  def dateOfBirth
}

athletes = athleteDAO.all('firstname').collect{ new
Athlete(it) }
```

Listing 10.5 Athlete example application layer: AthleteApplication

```
class AthleteApplication {
  def helper     = new DbHelper()
  def athleteDAO = new AthleteDAO(db: helper.db)
  def sortBy     = 'athleteId'

  def init() {
    helper.executeDdl(athleteDAO)
  }
  def exit() { System.exit(0) }

  def sort(field) {
    sortBy = field.join(',')
    list()
```

```
}
def create(List args) {
  athleteDAO.create(args)
  list()
}
def list() {
  def athletes = athleteDAO.all(sortBy)
  println athletes.size() + ' Athlete(s) in DB: '
  println 'id firstname  lastname     dateOfBirth'
  athletes.each { athlete ->
    println athlete.athleteId +': ' +
      athlete.firstname.padRight(10) + ' ' +
      athlete.lastname.padRight(12)  + ' ' +
      athlete.dateOfBirth
  }
}
def update(id, field, newValue){
  def count = athleteDAO.update(field, newValue, id)
  println count +' row(s) updated'
  list()
}
def delete(id) {
  def count = athleteDAO.delete(*id)
  println count +' row(s) deleted'
  list()
}
def mainLoop() {
  while(true) {
      println 'commands: create list update delete sort exit'
      def input = System.in.readLine().tokenize()
      def method = input.remove(0)
      invokeMethod(method, input)
    }
  }
}
app = new AthleteApplication()
app.init()
app.mainLoop()
```

六、 脚本集成

➔ Evaluating expressions and scripts with GroovyShell

```
def shell = new GroovyShell()
def result = shell.evaluate("12 + 23")
assert result == 35


// Java
import groovy.lang.GroovyShell;
public class HelloIntegrationWorld {
  public static void main(String[] args) {
    GroovyShell shell = new GroovyShell();
    Object result = shell.evaluate("12+23");
    assert new Integer(35).equals(result);
  }
}
```

```
Object evaluate(File file)
Object evaluate(InputStream in)
Object evaluate(InputStream in, String fileName)
Object evaluate(String scriptText)
Object evaluate(String scriptText, String fileName)
```

- 参数传递

```
def binding = new Binding()
binding.mass = 22.3
binding.velocity = 10.6
def shell = new GroovyShell(binding)
def expression = "mass * velocity ** 2 / 2"
assert shell.evaluate(expression) == 1252.814
binding.setVariable("mass", 25.4)
assert shell.evaluate(expression) == 1426.972
```

....

```
def binding = new Binding(x: 6, y: 4)
def shell = new GroovyShell(binding)
shell.evaluate('''
  xSquare = x * x
  yCube   = y * y * y
''')
assert binding.getVariable("xSquare") == 36
assert binding.yCube == 64
```

....

```
def binding = new Binding()
def shell = new GroovyShell(binding)
shell.evaluate('''
  def localVariable = "local variable"
  bindingVariable   = "binding variable"
''')
assert binding.getVariable("bindingVariable") == "binding
variable"
```

➔ **Generating dynamic classes at runtime**

```
def shell = new GroovyShell()
def clazz = shell.evaluate('''
   class MyClass {
      def method() { "value" }
   }
   return MyClass
''')
assert clazz.name == "MyClass"
def instance = clazz.newInstance()
assert instance.method() == "value"
```

解析脚本

```
def monthly = "amount*(rate/12) / (1-(1+rate/12)**-
numberOfMonths)"
def shell = new GroovyShell()
def script = shell.parse(monthly)
script.binding.amount = 154000
script.rate = 3.75/100
script.numberOfMonths = 240
assert script.run() == 913.0480050387338
script.binding = new Binding(amount: 185000,
                             rate: 3.50/100,
                             numberOfMonths: 300)
assert script.run() == 926.1536089487843
```

run 方法:

```
run(String script, String[] args)
run(File scriptFile, String scriptName, String[] args)
run(InputStream scriptStream, String scriptName, String[]
args)
```

GroovyShell 构造函数

```
public GroovyShell()
public GroovyShell(Binding binding)
public GroovyShell(Binding binding,
                   CompilerConfiguration config)
public GroovyShell(CompilerConfiguration config)
public GroovyShell(ClassLoader parent)
public GroovyShell(ClassLoader parent,
                   Binding binding)
public GroovyShell(ClassLoader parent,
                   Binding binding,
                   CompilerConfiguration config)
```

Table 11.2 The most useful methods in `CompilerConfiguration`

Method signature	Description
`setClasspath (String path)`	Define your own classpath used to look for classes, allowing you to restrict the application classpath and/or enhance it with other libraries
`setDebug (boolean debug)`	Set to true to get full, unfiltered stacktraces when exceptions are written on the error stream
`setOutput (PrintWriter writer)`	Set the writer compilation errors will be printed to
`setScriptBaseClass (String clazz)`	Define a subclass of `Script` as the base class for script instances
`setSourceEncoding (String enc)`	Set the encoding of the scripts to evaluate, which is important when parsing scripts from files or input streams that use a different encoding than the platform default
`setRecompileGroovySource (boolean b)`	Set to true to reload Groovy sources that have changed after they have been compiled—by default, this flag is set to false
`setMinimumRecompilationInterval (int millis)`	Set the minimum amount of time to wait before checking if the sources are more recent than the compiled classes

```
abstract class BaseScript extends Script {
   def multiply(a, b) { a * b }
}
def conf = new CompilerConfiguration()
conf.setScriptBaseClass("BaseScript")
def shell = new GroovyShell(conf)
def value = shell.evaluate('''
   multiply(5, 6)
''')
assert value == 30


// Java
MethodClosure mclos = new MethodClosure(multiplicator,
"multiply");
Binding binding = new Binding();
binding.setVariable("multiply", mclos);
GroovyShell shell = new GroovyShell(binding);
shell.evaluate("multiply(5, 6)");
```

➔ **Using the Groovy script engine**

```
def engine = new GroovyScriptEngine(".")
def engine = new GroovyScriptEngine([".", "../folder "])
def engine = new GroovyScriptEngine(
   ["file://.", "http://someUrl"]*.toURL() as URL[])
def engine = new GroovyScriptEngine(".", parentCL)

def engine = new GroovyScriptEngine(".")
def value  = engine.run("test/MyScript.groovy", new Binding())

def engine = new GroovyScriptEngine(".")
def clazz  = engine.loadScriptByName("test.MyScript")
def myResourceConnector = getResourceConnector()
def engine  = new GroovyScriptEngine(myResourceConnector)
def engine2 = new GroovyScriptEngine(myResourceConnector,
parent)

public URLConnection getResourceConnection(String name)
   throws ResourceException;

public long        getLastModified()
public URL         getURL()
public InputStream getInputStream() throws IOException
```

Parsing and loading Groovy classes

```
class Hello {
   def greeting() { "Hello!" }
}
def   gcl = new GroovyClassLoader()
Class greetingClass = gcl.parseClass(new File("Hello.groovy"))
assert "Hello!" == greetingClass.newInstance().greeting()




GroovyClassLoader gcl = new GroovyClassLoader();
Class greetingClass = gcl.parseClass(new
File("Hello.groovy"));
GroovyObject hello  = (GroovyObject)
greetingClass.newInstance();
Object[] args   = {};
assert "Hello!".equals(hello.invokeMethod("greeting", args));
```

Table 11.3 The sweet spots and limitations of the different integration mechanisms

Mechanism	Sweet spot	Limitations
`GroovyShell`	Perfect for single-line user input and small expressions Supports reloading Robust security available	Will not scale to dependent scripts
`GroovyScriptEngine`	Nice for dependent scripts Supports reloading	Does not support classes Does not support security
`GroovyClassLoader`	Most powerful integration mechanism Supports reloading Robust security available	Trickier to handle in the case of a complex classloader hierarchy
Spring scripting support	Integrates well with Spring Lets you switch languages easily Supports reloading	Requires Spring
JSR-223	Lets you switch languages easily	Requires Java 6 Does not support security Does not support reloading
Bean Scripting Framework	Lets you switch languages easily Doesn't require Java 6	Does not support security Does not support reloading More limited capabilities than JSR-223