

Table of Contents

Workspace Recipes	1
Using Workspaces	1
Using a Directory Workspace	3
Investigating Workspaces	4
Creating Workspaces	5

Workspace Recipes

The Workspace classes are in the [geoscript.workspace](#) package.

A Workspace is a collection of Layers. You can create, add, remove, and get Layers. There are many different kinds of Workspaces in GeoScript including Memory, PostGIS, Directory (for Shapefiles), GeoPackage, and many more.

Using Workspaces

Create a Workspace

```
Workspace workspace = new Workspace()
```

Create a Layer

```
Schema schema = new Schema("cities", [  
    new Field("geom", "Point", "EPSG:4326"),  
    new Field("id", "Integer"),  
    new Field("name", "String")  
)  
Layer layer = workspace.create(schema)  
println layer
```

```
cities
```

Check whether a Workspace has a Layer by name

```
boolean hasCities = workspace.has("cities")  
println hasCities
```

```
true
```

Get a Layer from a Workspace

```
Layer citiesLayer = workspace.get('cities')  
println citiesLayer
```

```
cities
```

Add a Layer to a Workspace

```
Schema statesSchema = new Schema("states", [  
    new Field("geom", "Polygon", "EPSG:4326"),  
    new Field("id", "Integer"),  
    new Field("name", "String")  
])  
Layer statesLayer = new Layer("states", statesSchema)  
workspace.add(statesLayer)  
println workspace.has("states")
```

true

Get the names of all Layers in a Workspace

```
List<String> names = workspace.names  
names.each { String name ->  
    println name  
}
```

MySQL (JNDI)
Directory of spatial files (shapefiles)
PostGIS (JNDI)
Properties
Spatialite (JNDI)
MySQL
GeoPackage
SQLite
Web Feature Server (NG)
Shapefile
H2
H2 (JNDI)
Geobuf
PostGIS

Remove a Layer from a Workspace

```
workspace.remove("cities")  
println workspace.has('cities')
```

false

Close the Workspace when you are done

```
workspace.close()
```

Using a Directory Workspace

A Directory Workspace is a directory of Shapefiles.

Create a Directory Workspace

```
Directory directory = new Directory("src/main/resources/data")
println directory.toString()
```

```
Directory[/home/travis/build/jericks/geoscript-groovy-
cookbook/src/main/resources/data]
```

View the Workspace's format

```
String format = directory.format
println format
```

```
Directory
```

View the Workspace's File

```
File file = directory.file
println file
```

```
/home/travis/build/jericks/geoscript-groovy-cookbook/src/main/resources/data
```

View the Workspace's list of Layer names

```
List names = directory.names
names.each { String name ->
    println name
}
```

```
states
```

Get a Layer by name

```
Layer layer = directory.get("states")
int count = layer.count
println "Layer ${layer.name} has ${count} Features."
```

Layer states has 49 Features.

Close the Directory when done.

```
directory.close()
```

Investigating Workspaces

Get available Workspace names

```
List<String> names = Workspace.getWorkspaceNames()
names.each { String name ->
    println name
}
```

MySQL (JNDI)
Directory of spatial files (shapefiles)
PostGIS (JNDI)
Properties
Spatialite (JNDI)
MySQL
GeoPackage
SQLite
Web Feature Server (NG)
Shapefile
H2
H2 (JNDI)
Geobuf
PostGIS

Get parameters for a Workspace

```
List<Map> parameters = Workspace.getWorkspaceParameters("GeoPackage")
parameters.each { Map param ->
    println "Parameter = ${param.key} Type = ${param.type} Required?
    ${param.required}"
}
```

```
Parameter = dbtype Type = java.lang.String Required? true
Parameter = database Type = java.io.File Required? true
Parameter = passwd Type = java.lang.String Required? false
Parameter = namespace Type = java.lang.String Required? false
Parameter = Expose primary keys Type = java.lang.Boolean Required? false
Parameter = fetch size Type = java.lang.Integer Required? false
Parameter = Batch insert size Type = java.lang.Integer Required? false
Parameter = Primary key metadata table Type = java.lang.String Required? false
Parameter = Session startup SQL Type = java.lang.String Required? false
Parameter = Session close-up SQL Type = java.lang.String Required? false
Parameter = Callback factory Type = java.lang.String Required? false
```

Creating Workspaces

Creating a Workspace from a connection string

You can create a Workspace from a connection string that contains parameters in key=value format with optional single quotes.

Create a Shapefile Workspace

```
String connectionString = "url='states.shp' 'create spatial index'=true"
Workspace workspace = Workspace.getWorkspace(connectionString)
```

Create a GeoPackage Workspace

```
connectionString = "dbtype=geopkg database=layers.gpkg"
workspace = Workspace.getWorkspace(connectionString)
```

Create a H2 Workspace

```
connectionString = "dbtype=h2 database=layers.db"
workspace = Workspace.getWorkspace(connectionString)
```

Creating a Workspace from a connection map

You can create a Workspace from a connection map that contains parameters.

Create a H2 Workspace

```
Map params = [dbtype: 'h2', database: 'test.db']
Workspace workspace = Workspace.getWorkspace(params)
```

Create a PostGIS Workspace

```
params = [  
  dbtype: 'postgis',  
  database: 'postgres',  
  host: 'localhost',  
  port: 5432,  
  user: 'postgres',  
  passwd: 'postgres'  
]  
workspace = Workspace.getWorkspace(params)
```

Create a GeoBuf Workspace

```
params = [file: 'layers.pbf', precision: 6, dimension: 2]  
workspace = Workspace.getWorkspace(params)
```

Creating Directory Workspaces

Create a Directory Workspace from a directory name

```
Workspace workspace = new Directory("src/main/resources/shapefiles")  
println workspace.format  
println "-----"  
workspace.names.each { String name ->  
  println "${name} (${workspace.get(name).count})"  
}
```

```
Directory  
-----  
ocean (2)  
countries (177)
```

Create a Directory Workspace from a File directory

```
Workspace workspace = new Directory(new File("src/main/resources/shapefiles"))  
println workspace.format  
println "-----"  
workspace.names.each { String name ->  
  println "${name} (${workspace.get(name).count})"  
}
```

Directory

ocean (2)

countries (177)

Create a Directory Workspace from a URL

```
Directory directory = Directory.fromURL(  
    new URL  
    ("http://www.naturalearthdata.com/http://www.naturalearthdata.com/download/110m/cultura  
l/ne_110m_admin_0_countries.zip"),  
    new File("naturalearth")  
)  
println directory.format  
println "-----"  
directory.names.each { String name ->  
    println "${name} (${directory.get(name).count})"  
}
```

Directory

ne_110m_admin_0_countries (177)

Creating GeoPackage Workspaces

Create a GeoPackage Workspace from a file name

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")  
println workspace.format  
println "-----"  
workspace.names.each { String name ->  
    println "${name} (${workspace.get(name).count})"  
}
```

GeoPackage

countries (177)

ocean (2)

places (326)

rivers (460)

states (52)

Create a GeoPackage Workspace from a File

```
Workspace workspace = new GeoPackage(new File("src/main/resources/data.gpkg"))
println workspace.format
println "-----"
workspace.names.each { String name ->
    println "${name} (${workspace.get(name).count})"
}
```

```
GeoPackage
-----
countries (177)
ocean (2)
places (326)
rivers (460)
states (52)
```

Creating H2 Workspaces

Create a H2 Workspace from a File

```
Workspace workspace = new H2(new File("src/main/resources/h2/data.db"))
println workspace.format
println "--"
workspace.names.each { String name ->
    println "${name} (${workspace.get(name).count})"
}
```

```
H2
--
countries (177)
ocean (2)
places (326)
states (52)
```

Creating Geobuf Workspaces

Create a Geobuf Workspace from a File

```
Workspace workspace = new Geobuf(new File("src/main/resources/geobuf"))
println workspace.format
println "-----"
workspace.names.each { String name ->
    println "${name} (${workspace.get(name).count})"
}
```

```
Geobuf
-----
ocean (2)
places (326)
countries (177)
```

Creating Property Workspaces

Create a Property Workspace from a File

```
Workspace workspace = new Property(new File("src/main/resources/property"))
println workspace.format
println "-----"
workspace.names.each { String name ->
    println "${name} (${workspace.get(name).count})"
}
```

```
Property
-----
circles (10)
places (10)
```