# Table of Contents

# 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
}
```

```
H2
PostGIS
PostGIS (JNDI)
MySQL
Properties
Shapefile
MBTiles with vector tiles
MySQL (JNDI)
Web Feature Server (NG)
Flatgeobuf
SQLite
Geobuf
H2 (JNDI)
GeoPackage
Directory of spatial files (shapefiles)
```

*Remove a Layer from a Workspace*

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

```
false
```

*Close the Workspace when you are done*

```
workspace.close()
```

# Creating an in Memory Workspace

*The empty Workspace constructor creates an in Memory Workspace.  You can create a Layer by passing a name and a list of Fields. You can then remove the Layer by passing a reference to the Layer.*

```
Workspace workspace = new Workspace()

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

workspace.remove(layer)
println workspace.has(layer.name)
```

```
cities
false
```

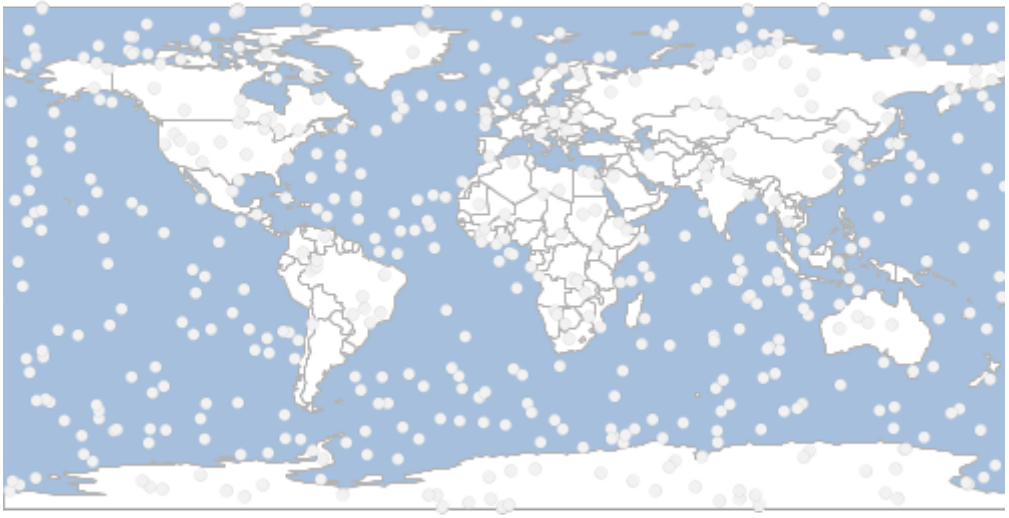
# Add Layer's Features in Chunks

*When adding a large Layer to a Workspace, you can add Features in chunks.*

```
Workspace workspace = new Memory()
Layer layer = workspace.create("points", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer")
])
Bounds bounds = new Bounds(-180,-90, 180,90, "EPSG:4326")
Geometry.createRandomPoints(bounds.geometry, 500).geometries.eachWithIndex { Geometry
geom, int i ->
    layer.add([geom: geom, id: i])
}
println "Original Layer has ${layer.count} features."

Layer copyOfLayer = workspace.add(layer, "random points", 100)
println "Copied Layer has ${copyOfLayer.count} features."
```

```
Original Layer has 500 features.
Copied Layer has 500 features.
```



## 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/runner/work/geoscript-groovy-cookbook/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/runner/work/geoscript-groovy-cookbook/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
}
```

```
H2
PostGIS
PostGIS (JNDI)
MySQL
Properties
Shapefile
MBTiles with vector tiles
MySQL (JNDI)
Web Feature Server (NG)
Flatgeobuf
SQLite
Geobuf
H2 (JNDI)
GeoPackage
Directory of spatial files (shapefiles)
```

*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
Parameter = read_only Type = java.lang.Boolean Required? false
Parameter = memory map size Type = java.lang.Integer Required? false
```

# Creating Workspaces

## Creating a Workspace from a connection string

You can create a Workspace from a connection string that contains paramters 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)
```

*You can use the withWorkspace method to automatically handle closing the Workspace.*

```
Workspace.withWorkspace("dbtype=geopkg database=src/main/resources/data.gpkg") {
Workspace workspace ->
    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 a Workspace from a connection map

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

*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)
```

*You can use the withWorkspace method to automatically handle closing the Workspace.*

```
Workspace.withWorkspace([dbtype: 'geopkg', database: 'src/main/resources/data.gpkg'])
{ Workspace workspace ->
    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 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)
```

*Create a H2 Workspace with basic parameters*

```
H2 h2 = new H2(
        "database",    ①
        "localhost",   ②
        "5421",        ③
        "geo",         ④
        "user",        ⑤
        "password"     ⑥
)
```

① Database name

② Host name

③ Port

④ User name

⑤ Password

*Create a H2 Workspace with named parameters.  Only the database name is required.*

```
H2 h2 = new H2("database",
        "host": "localhost",
        "port": "5412",
        "schema": "geo",
        "user": "user",
        "password": "secret"
)
```

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

## Creating Flatgeobuf Workspaces

*Create a Flatgeobuf Workspace from a File*

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

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

## 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)
```

## Creating SQLite Workspaces

*Create a SQLite Workspace from a File*

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

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

## Creating PostGIS Workspaces

*Create a PostGIS Workspace with basic parameters*

```
PostGIS postgis = new PostGIS(
        "database",    ①
        "localhost",   ②
        "5432",        ③
        "public",      ④
        "user",        ⑤
        "password"     ⑥
)
```

① Database name

② Host name

③ Port

④ Schema

⑤ User name

⑥ Password

*Create a PostGIS Workspace with advanced parameters*

```
PostGIS postgis = new PostGIS(
        "database",                   ①
        "localhost",                  ②
        "5432",                       ③
        "public",                     ④
        "user",                       ⑤
        "password",                   ⑥
        true,                         ⑦
        true,                         ⑧
        "OWNER geo TABLESPACE points" ⑨
)
```

① Database name

② Host name

③ Port

④ Schema

⑤ User name

⑥ Password

⑦ Estimated Extent

⑧ Create Database

⑨ Create Database Params

*Create a PostGIS Workspace with named parameters.  Only the database name is required.*

```
PostGIS postgis = new PostGIS("database",
        "host": "localhost",
        "port": "5432",
        "schema": "public",
        "user": "user",
        "password": "secret",
        "estimatedExtent": false,
        "createDatabase": false,
        "createDatabaseParams": "OWNER geo TABLESPACE points"
)
```

*Delete a PostGIS database.*

```
PostGIS.deleteDatabase(
        "database",    ①
        "localhost",   ②
        "5432",        ③
        "user",        ④
        "password"     ⑤
)
```

① Database name

② Host name

③ Port

④ User name

⑤ Password

*Delete a PostGIS database with named parameters.  Only the database name is required.*

```
PostGIS.deleteDatabase("database",
        "host": "localhost",
        "port": "5432",
        "user": "user",
        "password": "secret"
)
```

## Creating MySQL Workspaces

*Create a MySQL Workspace with basic parameters*

```
MySQL mysql = new MySQL(
        "database",    ①
        "localhost",   ②
        "3306",        ③
        "user",        ④
        "password"     ⑤
)
```

① Database name

② Host name

③ Port

④ User name

⑤ Password

*Create a MySQL Workspace with named parameters.  Only the database name is required.*

```
MySQL mysql = new MySQL("database",
        "host": "localhost",
        "port": "3306",
        "user": "user",
        "password": "secret"
)
```

## Creating SpatiaLite Workspaces

The SpatiaLite Workspace requires GDAL and OGR to be installed with Java support.

*Create a SpatiaLite Workspace with a File name*

```
SpatiaLite spatialite = new SpatiaLite("db.sqlite")
```

*Create a SpatiaLite Workspace with a File*

```
File directory = new File("databases")
File file = new File("db.sqlite")
SpatiaLite spatialite = new SpatiaLite(file)
```

## Creating WFS Workspaces

*Create a WFS Workspace with a URL*

```
WFS wfs = new WFS
("http://localhost:8080/geoserver/ows?service=wfs&version=1.1.0&request=GetCapabilitie
s")
```

## Creating OGR Workspaces

The OGR Workspace requires GDAL and OGR to be installed with Java support.

On Ubuntu, you can install GDAL and OGR with the following commands:

```
sudo apt-get install gdal-bin

sudo apt-get install libgdal-java
```

*Determine if OGR is available.*

```
boolean isAvailable = OGR.isAvailable()
```

*Get OGR Drivers.*

```
Set<String> drivers = OGR.drivers
```

*Get a Shapefile Workspace from OGR.*

```
File file = new File("states.shp")
OGR ogr = new OGR("ESRI Shapefile", file.absolutePath)
```

*Get a SQLite Workspace from OGR*

```
File file = new File("states.sqlite")
OGR ogr = new OGR("SQLite", file.absolutePath)
```

*Get a GeoJSON Workspace from OGR*

```
File file = new File("states.json")
OGR ogr = new OGR("GeoJSON", file.absolutePath)
```

# Database Workspace

## SQL

*Run SQL queries directly against Databse Workspace (PostGIS, MySQL, H2)*

```
Database workspace = new H2(new File("src/main/resources/h2/data.db"))
Sql sql = workspace.sql
```

*Count the number of results*

```
int numberOfPlaces = sql.firstRow("SELECT COUNT(*) as count FROM \"places\"").get
("count") as int
println "# of Places = ${numberOfPlaces}"
```

```
# of Places = 326
```

*Calculate statistics*

```
GroovyRowResult result = sql.firstRow("SELECT MIN(ELEVATION) as min_elev,
MAX(ELEVATION) as max_elev, AVG(ELEVATION) as avg_elev FROM \"places\"")
println "Mininum Elevation = ${result.get('min_elev')}"
println "Maximum Elevation = ${result.get('max_elev')}"
println "Average Elevation = ${result.get('avg_elev')}"
```

```
Mininum Elevation = 0.0
Maximum Elevation = 2320.0
Average Elevation = 30.085889570552148
```

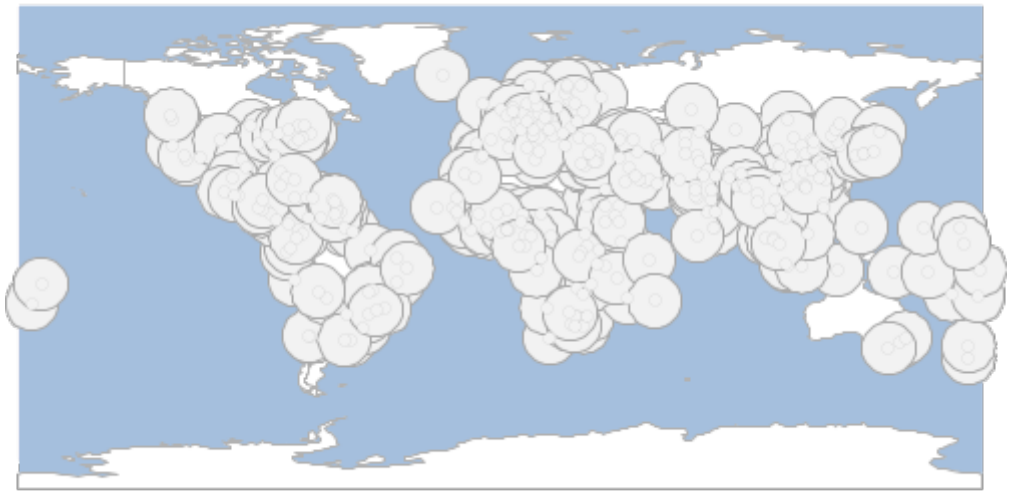
*Select rows*

```
List<String> names = []
sql.eachRow "SELECT TOP 10 \"NAME\" FROM \"places\" ORDER BY \"NAME\" DESC ", {
    names.add(it["NAME"])
}
names.each { String name ->
    println name
}
```

```
 Zürich
Zibo
Zhengzhou
Zagreb
Yerevan
Yaounde
Yamoussoukro
Xian
Wuhan
Windhoek
```

*Execute spatial sql*

```
Workspace memory = new Memory()
Layer layer = memory.create("places_polys", [new Field("buffer", "Polygon"), new
Field("name", "String")])
sql.eachRow "SELECT ST_Buffer(\"the_geom\", 10) as buffer, \"NAME\" as name FROM
\"places\"", {row ->
    Geometry poly = Geometry.fromWKB(row.buffer as byte[])
    layer.add([buffer: poly, name: row.NAME])
}
```



## View

*Create a new Layer from a SQL View*

```
Database workspace = new H2(new File("src/main/resources/h2/data.db"))
Layer layer = workspace.createView(
        "megacities",                                                  ①
        "SELECT * FROM \"places\" WHERE \"MEGACITY\" = '%mega%'", ②
        new Field("the_geom","Point","EPSG:4326"),                     ③
        params: [['mega', '1']]                                        ④
)
boolean hasLayer1 = workspace.has("megacities")
println "Does layer exist? ${hasLayer1}"
```

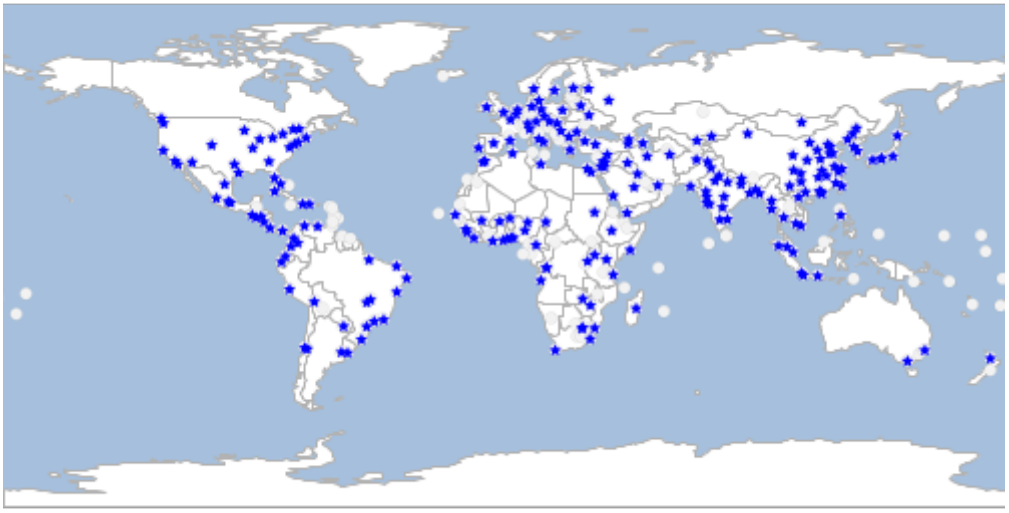
① The layer name

② The SQL Statement

③ The Geometry Field

④ Query Parameters

```
Does layer exist? true
```



*Remove the new Layer created from a SQL View*

```
workspace.deleteView("megacities")
boolean hasLayer2 = workspace.has("megacities")
println "Does layer exist? ${hasLayer2}"
```

```
Does layer exist? false
```

## Index

*Create an Index*

```
Database workspace = new H2(new File("src/main/resources/h2/data.db"))
workspace.createIndex("places","name_idx","NAME",true)
workspace.createIndex("places","megacity_idx","MEGACITY",false)
workspace.createIndex("places","a3_idx", ["SOV_A3", "ADM0_A3"],false)
```

*Get an Index*

```
List<Map> indexes = workspace.getIndexes("places")
indexes.each { Map index ->
    println "Index name = ${index.name}, unique = ${index.unique}, attributes =
${index.attributes}"
}
```

```
Index name = PRIMARY_KEY_C, unique = true, attributes = [fid]
Index name = name_idx, unique = true, attributes = [NAME]
Index name = a3_idx, unique = false, attributes = [SOV_A3, ADM0_A3]
Index name = megacity_idx, unique = false, attributes = [MEGACITY]
```

*Remove an Index*

```
workspace.deleteIndex("places", "name_idx")
workspace.deleteIndex("places", "megacity_idx")
workspace.deleteIndex("places", "a3_idx")
```