

Table of Contents

Layer Recipes	1
Getting a Layer's Properties	1
Getting a Layer's Features	2
Adding, Updating, and Deleting	9
Shapefiles	16
Property	19
Geoprocessing	22
Layer Algebra	29
Reading and Writing Features	36
Graticules	45

Layer Recipes

The Layer classes are in the [geoscript.layer](#) package.

A Layer is a collection of Features.

Getting a Layer's Properties

Get a Layer from a Workspace and it's name

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("countries")
String name = layer.name
println "Name: ${name}"
```

Name: countries

The Layer's Format

```
String format = layer.format
println "Format: ${format}"
```

Format: GeoPackage

Count the number of Features

```
int count = layer.count
println "# of Features: ${count}"
```

of Features: 177

Get the Layer's Projection

```
Projection proj = layer.proj
println "Projection: ${proj}"
```

Projection: EPSG:4326

Get the Bounds of the Layer

```
Bounds bounds = layer.bounds
println "Bounds: ${bounds}"
```

```
Bounds: (-179.99999999999997, -
90.000000000000003, 180.00000000000014, 83.64513000000002, EPSG:4326)
```

Getting a Layer's Features

Each Feature

Iterate over a Layer's Features

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
layer.eachFeature { Feature feature ->
    println feature["NAME_1"]
}
```

```
Minnesota
Montana
North Dakota
Hawaii
Idaho
Washington
Arizona
California
Colorado
Nevada
...
```

Iterate over a subset of a Layer's Features

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
layer.eachFeature("NAME_1 LIKE 'M%'") { Feature feature ->
    println feature["NAME_1"]
}
```

Minnesota
Montana
Missouri
Massachusetts
Mississippi
Maryland
Maine
Michigan

Iterate over a Layer's Features with parameters.

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
layer.eachFeature(sort: ["NAME_1"], start: 0, max: 5, fields: ["NAME_1"], filter:
"NAME_1 LIKE 'M%') { Feature feature ->
    println feature["NAME_1"]
}
```

Maine
Maryland
Massachusetts
Michigan
Minnesota

Parameters

- filter: The Filter or Filter String to limit the Features. Defaults to null.
- sort: A List of Lists that define the sort order [[Field or Field name, "ASC" or "DESC"],...]. Not all Layers support sorting!
- max: The maximum number of Features to include
- start: The index of the record to start the cursor at. Together with maxFeatures this simulates paging. Not all Layers support the start index and paging!
- fields: A List of Fields or Field names to include. Used to select only a subset of Fields.

Read all Features

Read all Feature into a List

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
List<Feature> features = layer.features

println "# Features = ${features.size()}"
features.each { Feature feature ->
    println feature["NAME_1"]
}
```

```
# Features = 52
Minnesota
Montana
North Dakota
Hawaii
Idaho
Washington
Arizona
California
Colorado
Nevada
...
```

Collect values

Collect values from a Layer's Features

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
List<String> names = layer.collectFromFeature { Feature f ->
    f["NAME_1"]
}.sort()

println "# Names = ${names.size()}"
names.each { String name ->
    println name
}
```

```
# Names = 52
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
...
```

Collect values from a Layer's Features with parameters.

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
List<String> names = layer.collectFromFeature(
    sort: ["NAME_1"],
    start: 0,
    max: 5,
    fields: ["NAME_1"],
    filter: "NAME_1 LIKE 'M%'" ) { Feature f ->
        f["NAME_1"]
    }

println "# Names = ${names.size()}"
names.each { String name ->
    println name
}
```

```
# Names = 5
Maine
Maryland
Massachusetts
Michigan
Minnesota
```

First

Get the first Feature that matches the Filter.

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
Feature feature = layer.first(filter: "NAME_1='Washington'")
println feature.get("NAME_1")
```

Washington



Get the first Feature sorted by name ascending and descending.

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")

Feature featureAsc = layer.first(sort: "NAME_1 ASC")
println featureAsc.get("NAME_1")

Feature featureDesc = layer.first(sort: "NAME_1 DESC")
println featureDesc.get("NAME_1")
```

Alabama
Wyoming

Filter



Create a new Layer from an existing Layer with just the Features that match a Filter.

```
Workspace workspace = new Directory("target")
Layer layer = workspace.get("countries")
Layer disputedLayer = layer.filter("TYPE='Disputed'")
```



Cursor

Iterate over a Layer's Features with a Cursor

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")
Layer layer = workspace.get("states")
Cursor cursor = layer.cursor
cursor.each { Feature feature ->
    println feature["NAME_1"]
}
```



```
Minnesota  
Montana  
North Dakota  
Hawaii  
Idaho  
Washington  
Arizona  
California  
Colorado  
Nevada  
...
```

Iterate over a subset of a Layer's Features with a Cursor

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")  
Layer layer = workspace.get("states")  
Cursor cursor = layer.getCursor(filter: "NAME_1 LIKE 'M%'")  
while(cursor.hasNext()) {  
    Feature feature = cursor.next()  
    println feature["NAME_1"]  
}
```

```
Minnesota  
Montana  
Missouri  
Massachusetts  
Mississippi  
Maryland  
Maine  
Michigan
```

Iterate over a Layer's Features with parameters with a Cursor

```
Workspace workspace = new GeoPackage("src/main/resources/data.gpkg")  
Layer layer = workspace.get("states")  
layer.getCursor(sort: ["NAME_1"], start: 0, max: 5, fields: ["NAME_1"], filter:  
"NAME_1 LIKE 'M%'").each { Feature feature ->  
    println feature["NAME_1"]  
}
```

```
Maine  
Maryland  
Massachusetts  
Michigan  
Minnesota
```

Parameters

- filter: The Filter or Filter String to limit the Features. Defaults to null.
- sort: A List of Lists that define the sort order [[Field or Field name, "ASC" or "DESC"],...]. Not all Layers support sorting!
- max: The maximum number of Features to include
- start: The index of the record to start the cursor at. Together with maxFeatures this simulates paging. Not all Layers support the start index and paging!
- fields: A List of Fields or Field names to include. Used to select only a subset of Fields.

Adding, Updating, and Deleting

Add Features to a Layer

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

// Add a Feature with a Map
Map attributes = [
    geom: new Point(-122.333056, 47.609722),
    id: 1,
    name: "Seattle",
    state: "WA"
]
layer.add(attributes)

// Add a Feature with a List
List values = [
    new Point(-122.459444, 47.241389),
    2,
    "Tacoma",
    "WA"
]
layer.add(values)

// Add a Feature
Feature feature = schema.feature([
    id: 3,
    name: "Fargo",
    state: "ND",
    geom: new Point(-96.789444, 46.877222)
])
```

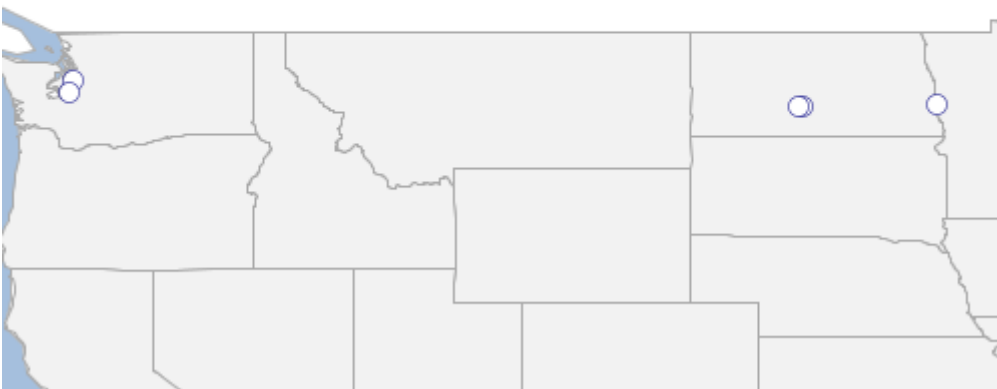
```

layer.add(feature)

// Add Features from a List of Maps
List<Map> features = [
    [
        geom: new Point(-100.778889, 46.813333),
        id: 4,
        name: "Bismarck",
        state: "ND"
    ],
    [
        geom: new Point(-100.891111, 46.828889),
        id: 5,
        name: "Mandan",
        state: "ND"
    ]
]
layer.add(features)

```

id	name	state
1	Seattle	WA
2	Tacoma	WA
3	Fargo	ND
4	Bismarck	ND
5	Mandan	ND



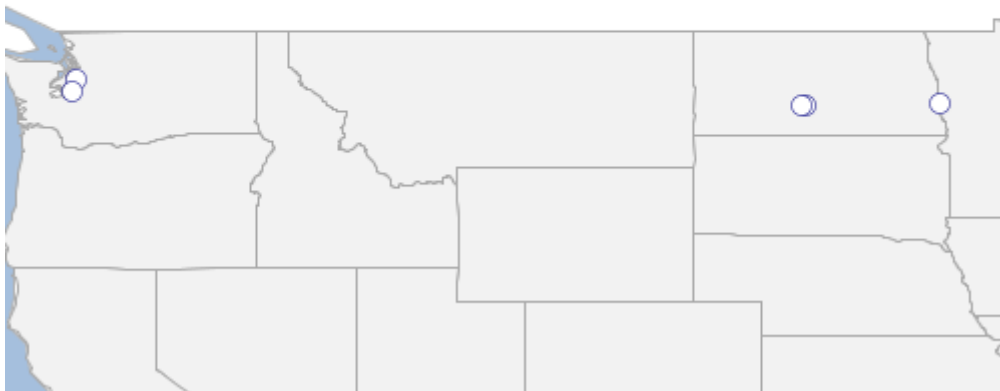
Update Features in a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String"),
    new Field("state", "String")
])
Layer layer = workspace.create(schema)
List<Map> features = [
    [
        geom: new Point(-122.333056, 47.609722),
        id: 1,
        name: "Seattle",
        state: "WA"
    ],
    [
        geom: new Point(-122.459444, 47.241389),
        id: 2,
        name: "Tacoma",
        state: "WA"
    ],
    [
        id: 3,
        name: " Fargo",
        state: "ND",
        geom: new Point(-96.789444, 46.877222)
    ],
    [
        geom: new Point(-100.778889, 46.813333),
        id: 4,
        name: "Bismarck",
        state: "ND"
    ],
    [
        geom: new Point(-100.891111, 46.828889),
        id: 5,
        name: "Mandan",
        state: "ND"
    ]
]
layer.add(features)

layer.update(layer.schema.state, "North Dakota", "state='ND'")
layer.update(layer.schema.state, "Washington", "state='WA'")
```

id	name	state
1	Seattle	Washington

id	name	state
2	Tacoma	Washington
3	Fargo	North Dakota
4	Bismarck	North Dakota
5	Mandan	North Dakota



Delete Features from a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String"),
    new Field("state", "String")
])
Layer layer = workspace.create(schema)
List<Map> features = [
    [
        geom: new Point(-122.333056, 47.609722),
        id: 1,
        name: "Seattle",
        state: "WA"
    ],
    [
        geom: new Point(-122.459444, 47.241389),
        id: 2,
        name: "Tacoma",
        state: "WA"
    ],
    [
        id:3,
        name: " Fargo",
        state: "ND",
        geom: new Point(-96.789444, 46.877222)
    ],
    [
        geom: new Point(-100.778889, 46.813333),
        id:4,
        name: "Bismarck",
        state: "ND"
    ],
    [
        geom: new Point(-100.891111, 46.828889),
        id: 5,
        name: "Mandan",
        state: "ND"
    ]
]
layer.add(features)

layer.delete("state='ND'")
```

id	name	state
1	Seattle	WA
2	Tacoma	WA

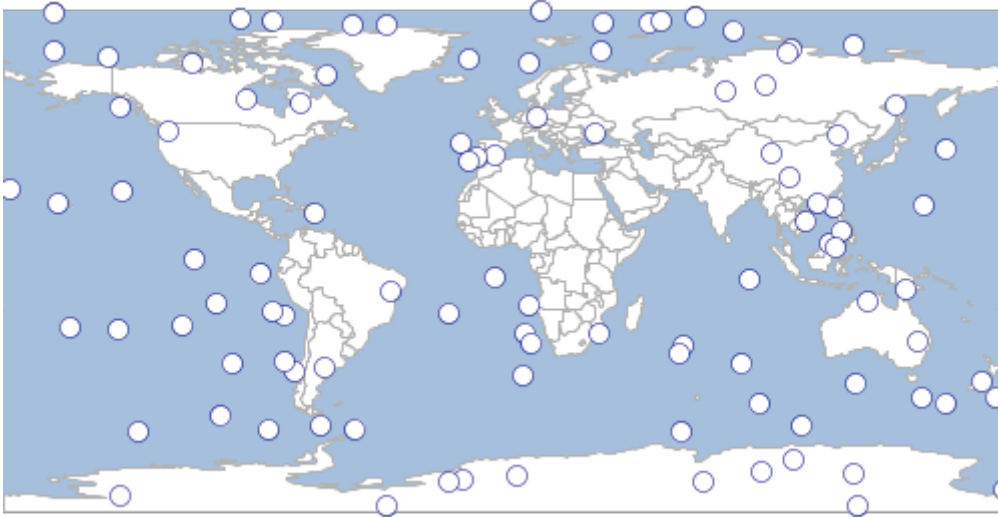


Add Features to a Layer using a Writer. A Writer can add Features more effeciently because is commits batches of Features in Transactions .

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

Bounds bounds = new Bounds(-180,-90,180,90, "EPSG:4326")
MultiPoint points = Geometry.createRandomPoints(bounds.geometry, 100)

geoscript.layer.Writer writer = layer.writer
try {
    points.geometries.eachWithIndex { Point point, int index ->
        Feature feature = writer.newFeature
        feature['id'] = index
        feature['geom'] = point
        writer.add(feature)
    }
} finally {
    writer.close()
}
```

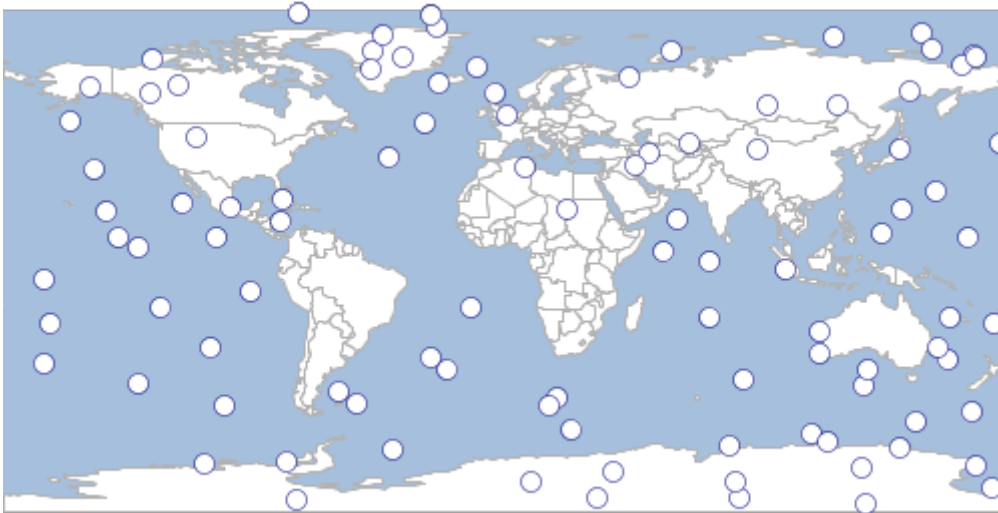


Add Features to a Layer using a Writer inside of a Closure.

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

Bounds bounds = new Bounds(-180, -90, 180, 90, "EPSG:4326")
MultiPoint points = Geometry.createRandomPoints(bounds.geometry, 100)

layer.withWriter { geoscript.layer.Writer writer ->
    points.geometries.eachWithIndex { Point point, int index ->
        Feature feature = writer.newFeature
        feature['id'] = index
        feature['geom'] = point
        writer.add(feature)
    }
}
```

Shapefiles

Shapefiles are a very commonly used format for storing spatial data. So, instead of creating a Directory Workspace and getting the Layer from the Workspace, you can use the Shapefile class.

Read

Read existing Shapefiles

```
Shapefile countries = new Shapefile("src/main/resources/shapefiles/countries.shp")
println "# Features in Countries = ${countries.count}"

Shapefile ocean = new Shapefile(new File("src/main/resources/shapefiles/ocean.shp"))
println "# Features in Ocean = ${ocean.count}"
```

```
# Features in Countries = 177
```

```
# Features in Ocean = 2
```



Create

Create a new Shapefile

```
Directory workspace = new Directory("target")
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String"),
    new Field("state", "String")
])
Layer layer = workspace.create(schema)
List<Map> features = [
    [
        geom: new Point(-122.333056, 47.609722),
        id: 1,
        name: "Seattle",
        state: "WA"
    ],
    [
        geom: new Point(-122.459444, 47.241389),
        id: 2,
        name: "Tacoma",
        state: "WA"
    ],
    [
        id: 3,
        name: " Fargo",
        state: "ND",
        geom: new Point(-96.789444, 46.877222)
    ],
    [
        geom: new Point(-100.778889, 46.813333),
        id: 4,
        name: "Bismarck",
        state: "ND"
    ],
    [
        geom: new Point(-100.891111, 46.828889),
        id: 5,
        name: "Mandan",
        state: "ND"
    ]
]
layer.add(features)
```

id	name	state
1	Seattle	WA
2	Tacoma	WA
3	Fargo	ND

id	name	state
4	Bismarck	ND
5	Mandan	ND



Property

GeoScript can store spatial data in a simple plain text format. With the Property class you can access a single property file directly.

Read

Read existing Property files

```
geoscript.layer.Property circles = new geoscript.layer.Property
("src/main/resources/property/circles.properties")
println "# Features in circles = ${circles.count}"

geoscript.layer.Property places = new geoscript.layer.Property(new File
("src/main/resources/property/places.properties"))
println "# Features in places = ${places.count}"
```

```
# Features in circles = 10
```

```
# Features in places = 10
```



Create

Create a new Property

```
geoscript.workspace.Property workspace = new geoscript.workspace.Property("target")
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String"),
    new Field("state", "String")
])
Layer layer = workspace.create(schema)
List<Map> features = [
    [
        geom: new Point(-122.333056, 47.609722),
        id: 1,
        name: "Seattle",
        state: "WA"
    ],
    [
        geom: new Point(-122.459444, 47.241389),
        id: 2,
        name: "Tacoma",
        state: "WA"
    ],
    [
        id: 3,
        name: " Fargo",
        state: "ND",
        geom: new Point(-96.789444, 46.877222)
    ],
    [
        geom: new Point(-100.778889, 46.813333),
        id: 4,
        name: "Bismarck",
        state: "ND"
    ],
    [
        geom: new Point(-100.891111, 46.828889),
        id: 5,
        name: "Mandan",
        state: "ND"
    ]
]
layer.add(features)
```

id	name	state
1	Seattle	WA
2	Tacoma	WA
3	Fargo	ND

id	name	state
4	Bismarck	ND
5	Mandan	ND



Geoprocessing

Reproject

Reproject a Layer from it's source projection to a target projection

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer states = workspace.get("states")
println "States = ${states.proj}"

Projection projection = new Projection("EPSG:3857")
Workspace outputWorkspace = new Memory()
Layer statesInWebMercator = states.reproject(projection, outputWorkspace,
"states_3857")
println "Reprojected States = ${statesInWebMercator.proj}"
```

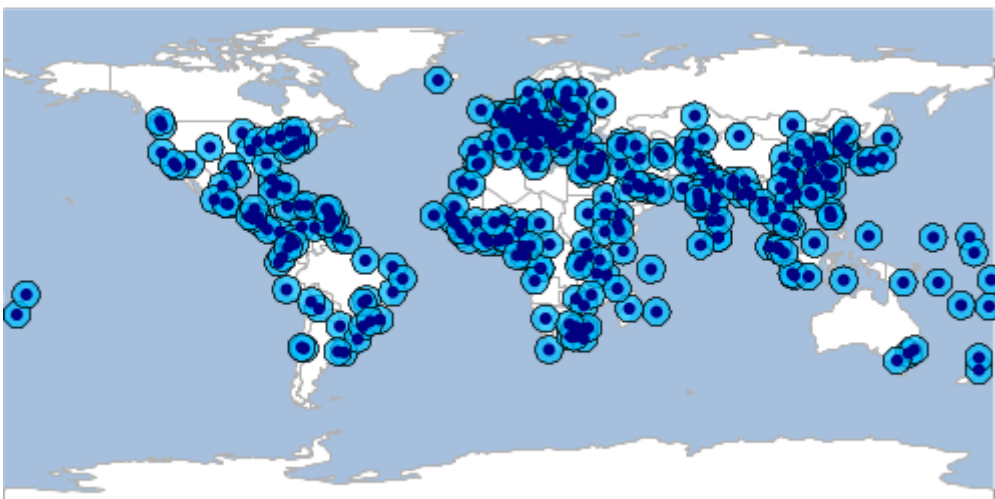
```
States = EPSG:4326
Reprojected States = EPSG:3857
```



Buffer

Buffer a Layer of populated places

```
Workspace workspace = new GeoPackage(new File("src/main/resources/data.gpkg"))  
Layer places = workspace.get("places")  
Layer buffer = places.buffer(5)
```



Dissolve

Dissolve a Layer by a Field

```
Workspace workspace = new Directory(new File("src/main/resources/data"))
Layer states = workspace.get("states")
Layer regions = states.dissolve(states.schema.get("SUB_REGION"))
```



Merge

Merge two Layer together

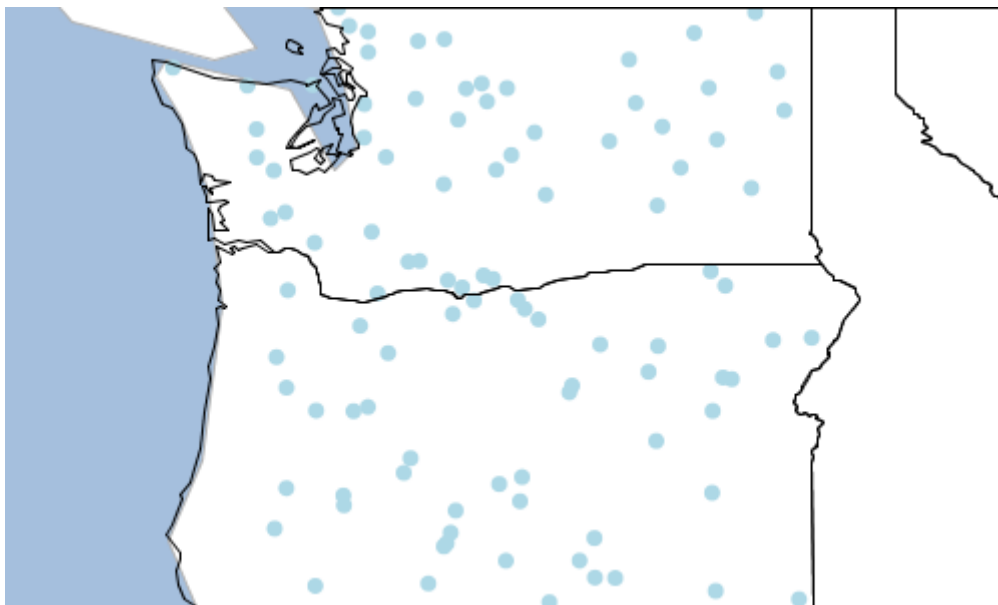
```
Workspace workspace = new Directory(new File("src/main/resources/data"))
Layer states = workspace.get("states")
Feature washington = states.first(filter: "STATE_NAME='Washington'")
Feature oregon = states.first(filter: "STATE_NAME='Oregon'")

Schema waSchema = new Schema("washington",[
    new Field("geom","Point","EPSG:4326"),
    new Field("id","int")
])
Layer waLayer = new Memory().create(waSchema)
waLayer.withWriter { geoscript.layer.Writer writer ->
    Geometry.createRandomPoints(washington.geom, 50).points.eachWithIndex { Point pt,
int index ->
        writer.add(waSchema.feature([geom: pt, id: index]))
    }
}
println "The Washington Layer has ${waLayer.count} features"

Schema orSchema = new Schema("oregon",[
    new Field("geom","Point","EPSG:4326"),
    new Field("id","int")
])
Layer orLayer = new Memory().create(orSchema)
orLayer.withWriter { geoscript.layer.Writer writer ->
    Geometry.createRandomPoints(oregon.geom, 50).points.eachWithIndex { Point pt, int
index ->
        writer.add(orSchema.feature([geom: pt, id: index]))
    }
}
println "The Oregon Layer has ${orLayer.count} features"

Layer mergedLayer = orLayer.merge(waLayer)
println "The merged Layer has ${mergedLayer.count} features"
```

```
The Washington Layer has 50 features
The Oregon Layer has 50 features
The merged Layer has 100 features
```



Split

Split a Layer into Layers based on the value from a Field

```
Workspace workspace = new GeoPackage(new File("src/main/resources/data.gpkg"))
Layer rivers = workspace.get("rivers")
Workspace outWorkspace = new Memory()
rivers.split(rivers.schema.get("scalerank"), outWorkspace)

outWorkspace.layers.each { Layer layer ->
    println "${layer.name} has ${layer.count} features"
}
```

```
rivers_0 has 1 features
rivers_1 has 27 features
rivers_2 has 35 features
rivers_3 has 53 features
rivers_4 has 71 features
rivers_5 has 67 features
rivers_6 has 206 features
```



Split a Layer into Layers based on another Layer

```
Schema schema = new Schema("grid",[
    new Field("geom","Polygon","EPSG:4326"),
    new Field("col","int"),
    new Field("row","int"),
    new Field("row_col","string")
])
Workspace gridWorkspace = new Directory("target")
Layer gridLayer = gridWorkspace.create(schema)
new Bounds(-180,-90,180,90,"EPSG:4326").generateGrid(2, 3, "polygon", {cell, col, row
->
    gridLayer.add([
        "geom": cell,
        "col": col,
        "row": row,
        "row_col": "${row} ${col}"
    ])
})

Workspace workspace = new GeoPackage(new File("src/main/resources/data.gpkg"))
Layer countries = workspace.get("countries")

Workspace outWorkspace = new Memory()
countries.split(gridLayer,gridLayer.schema.get("row_col"),outWorkspace)

outWorkspace.layers.each { Layer layer ->
    println "${layer.name} has ${layer.count} features"
}
```

```
countries_1_1 has 6 features
countries_1_2 has 6 features
countries_2_1 has 44 features
countries_2_2 has 74 features
countries_3_1 has 13 features
countries_3_2 has 70 features
```



Transform

Transform one Layer into another Layer

```
Workspace workspace = new Directory(new File("src/main/resources/data"))
Layer states = workspace.get("states")
Layer centroids = states.transform("centroids", [
    geom: "centroid(the_geom)",
    name: "strToUpperCase(STATE_NAME)",
    ratio: "FEMALE / MALE"
])
centroids.eachFeature(max: 5) {Feature f ->
    println "${f.geom} ${f['name']} = ${f['ratio']}"
}
```

```
ILLINOIS = 1.0587396098110435
DISTRICT OF COLUMBIA = 1.1447503268897763
DELAWARE = 1.0626439771122835
WEST VIRGINIA = 1.0817203227723509
MARYLAND = 1.0621588832568312
```



Layer Algebra

GeoScript can do layer algebra. All of the examples below use Layer A (red) and Layer B (green).



Clip

Clip Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.clip(layerB)
```



Clip Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.clip(layerA, outWorkspace, outLayer: "ba_clip")
```



Erase

Erase Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.erase(layerB)
```



Erase Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.erase(layerA, outWorkspace, outLayer: "ba_erase")
```



Identity

Identity Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.identity(layerB)
```




Identity Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.identity(layerA, outWorkspace: outWorkspace, outLayer:
"ba_identity")
```



Intersection

Intersection Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.intersection(layerB)
```



Intersection Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.intersection(layerA, outWorkspace: outWorkspace, outLayer:
"ba_intersection")
```



Symmetric Difference

Symmetric Difference Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.symDifference(layerB)
```



Symmetric Difference Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.symDifference(layerA, outWorkspace: outWorkspace, outLayer:
"ba_symdifference")
```



Update

Update Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.update(layerB)
```



Update Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.update(layerA, outWorkspace: outWorkspace, outLayer:
"ba_update")
```



Union

Union Layer A with Layer B

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Layer layerC = layerA.union(layerB)
```



Union Layer B with Layer A

```
Workspace workspace = new GeoPackage(new File("src/main/resources/layeralgebra.gpkg"))
Layer layerA = workspace.get("a")
Layer layerB = workspace.get("b")
Workspace outWorkspace = new Directory("target")
Layer layerC = layerB.union(layerA, outWorkspace, outLayer: "ba_union")
```



Reading and Writing Features

The Layer IO classes are in the [geoscript.layer.io](#) package.

Finding Layer Writer and Readers

List all Layer Writers

```
List<Writer> writers = Writers.list()
writers.each { Writer writer ->
    println writer.class.simpleName
}
```

```
CsvWriter
GeobufWriter
GeoJSONWriter
GeoRSSWriter
GmlWriter
GpxWriter
KmlWriter
MvtWriter
```

Find a Layer Writer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String")
])
Layer layer = workspace.create(schema)
layer.add([
    geom: new Point(-122.3204, 47.6024),
    id: 1,
    name: "Seattle"
])
layer.add([
    geom: new Point(-122.48416, 47.2619),
    id: 2,
    name: "Tacoma"
])

Writer writer = Writers.find("csv")
String csv = writer.write(layer)
println csv
```

```
"geom:Point:EPSG:4326","id:Integer","name:String"
"POINT (-122.3204 47.6024)","1","Seattle"
"POINT (-122.48416 47.2619)","2","Tacoma"
```

List all Layer Readers

```
List<Reader> readers = Readers.list()
readers.each { Reader reader ->
    println reader.class.simpleName
}
```

```
CsvReader
GeobufReader
GeoJSONReader
GeoRSSReader
GmlReader
GpxReader
KmlReader
MvtReader
```

Find a Layer Reader

```
Reader reader = Readers.find("csv")
Layer layer = reader.read("""geom:Point:EPSG:4326","id:Integer","name:String"
"POINT (-122.3204 47.6024)","1","Seattle"
"POINT (-122.48416 47.2619)","2","Tacoma"
""")
println "# features = ${layer.count}"
```

```
# features = 2
```

GeoJSON

Get GeoJSON String from a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String")
])
Layer layer = workspace.create(schema)
layer.add([
    geom: new Point(-122.3204, 47.6024),
    id: 1,
    name: "Seattle"
])
layer.add([
    geom: new Point(-122.48416, 47.2619),
    id: 2,
    name: "Tacoma"
])

String geojson = layer.toJSONString()
println geojson
```



```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          -122.3204,
          47.6024
        ]
      },
      "properties": {
        "id": 1,
        "name": "Seattle"
      },
      "id": "fid-6849e75_1633dc2de74_-693d"
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          -122.4842,
          47.2619
        ]
      },
      "properties": {
        "id": 2,
        "name": "Tacoma"
      },
      "id": "fid-6849e75_1633dc2de74_-693b"
    }
  ]
}

```

GeoBuf

Get GeoBuf String from a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String")
])
Layer layer = workspace.create(schema)
layer.add([
    geom: new Point(-122.3204, 47.6024),
    id: 1,
    name: "Seattle"
])
layer.add([
    geom: new Point(-122.48416, 47.2619),
    id: 2,
    name: "Tacoma"
])

String geobuf = layer.toGeobufString()
println geobuf
```

```
0a0269640a046e616d6510021806223d0a1d0a0c08001a089fd8d374c0ebb22d6a0218016a090a07536561
74746c650a1c0a0c08001a08ff6e77498a3892d6a0218026a080a065461636f6d61
```

GML

Get GML String from a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String")
])
Layer layer = workspace.create(schema)
layer.add([
    geom: new Point(-122.3204, 47.6024),
    id: 1,
    name: "Seattle"
])
layer.add([
    geom: new Point(-122.48416, 47.2619),
    id: 2,
    name: "Tacoma"
])

String gml = layer.toGMLString()
println gml
```

```
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs">
  <gml:boundedBy xmlns:gml="http://www.opengis.net/gml">
    <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:coord>
        <gml:X>
          -122.48416
        </gml:X>
        <gml:Y>
          47.2619
        </gml:Y>
      </gml:coord>
      <gml:coord>
        <gml:X>
          -122.3204
        </gml:X>
        <gml:Y>
          47.6024
        </gml:Y>
      </gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember xmlns:gml="http://www.opengis.net/gml">
    <gsf:cities xmlns:gsf="http://geoscript.org/feature" fid="fid-6849e75_1633dc2de74_-73c1">
      <gml:name>
        Seattle
      </gml:name>
    </gsf:cities>
  </gml:featureMember>
</wfs:FeatureCollection>
```

```

    <gsf:geom>
      <gml:Point>
        <gml:coord>
          <gml:X>
            -122.3204
          </gml:X>
          <gml:Y>
            47.6024
          </gml:Y>
        </gml:coord>
      </gml:Point>
    </gsf:geom>
    <gsf:id>
      1
    </gsf:id>
  </gsf:cities>
</gml:featureMember>
<gml:featureMember xmlns:gml="http://www.opengis.net/gml">
  <gsf:cities xmlns:gsf="http://geoscript.org/feature" fid="fid-
6849e75_1633dc2de74_-73bf">
    <gml:name>
      Tacoma
    </gml:name>
    <gsf:geom>
      <gml:Point>
        <gml:coord>
          <gml:X>
            -122.48416
          </gml:X>
          <gml:Y>
            47.2619
          </gml:Y>
        </gml:coord>
      </gml:Point>
    </gsf:geom>
    <gsf:id>
      2
    </gsf:id>
  </gsf:cities>
</gml:featureMember>
</wfs:FeatureCollection>

```

KML

Get KML String from a Layer

```
Workspace workspace = new Memory()
Schema schema = new Schema("cities", [
    new Field("geom", "Point", "EPSG:4326"),
    new Field("id", "Integer"),
    new Field("name", "String")
])
Layer layer = workspace.create(schema)
layer.add([
    geom: new Point(-122.3204, 47.6024),
    id: 1,
    name: "Seattle"
])
layer.add([
    geom: new Point(-122.48416, 47.2619),
    id: 2,
    name: "Tacoma"
])

String kml = layer.toKMLString()
println kml
```

```
<kml:kml xmlns:kml="http://www.opengis.net/kml/2.2">
  <kml:Document>
    <kml:Folder>
      <kml:name>
        cities
      </kml:name>
      <kml:Schema kml:name="cities" kml:id="cities">
        <kml:SimpleField kml:name="id" kml:type="Integer"/>
        <kml:SimpleField kml:name="name" kml:type="String"/>
      </kml:Schema>
      <kml:Placemark>
        <kml:name>
          fid-6849e75_1633dc2de74_-6a3f
        </kml:name>
        <kml:Style>
          <kml:IconStyle>
            <kml:color>
              ff0000ff
            </kml:color>
          </kml:IconStyle>
        </kml:Style>
        <kml:ExtendedData>
          <kml:SchemaData kml:schemaUrl="#cities">
            <kml:SimpleData kml:name="id">
              1
            </kml:SimpleData>
            <kml:SimpleData kml:name="name">
```

```

        Seattle
      </kml:SimpleData>
    </kml:SchemaData>
  </kml:ExtendedData>
  <kml:Point>
    <kml:coordinates>
      -122.3204,47.6024
    </kml:coordinates>
  </kml:Point>
</kml:Placemark>
<kml:Placemark>
  <kml:name>
    fid-6849e75_1633dc2de74_-6a3d
  </kml:name>
  <kml:Style>
    <kml:IconStyle>
      <kml:color>
        ff0000ff
      </kml:color>
    </kml:IconStyle>
  </kml:Style>
  <kml:ExtendedData>
    <kml:SchemaData kml:schemaUrl="#cities">
      <kml:SimpleData kml:name="id">
        2
      </kml:SimpleData>
      <kml:SimpleData kml:name="name">
        Tacoma
      </kml:SimpleData>
    </kml:SchemaData>
  </kml:ExtendedData>
  <kml:Point>
    <kml:coordinates>
      -122.48416,47.2619
    </kml:coordinates>
  </kml:Point>
</kml:Placemark>
</kml:Folder>
</kml:Document>
</kml:kml>

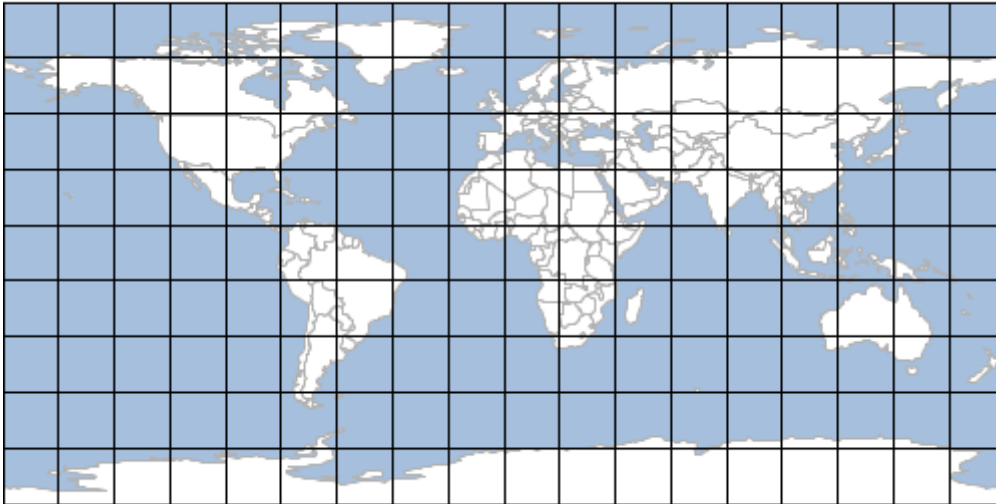
```

Graticules

Square

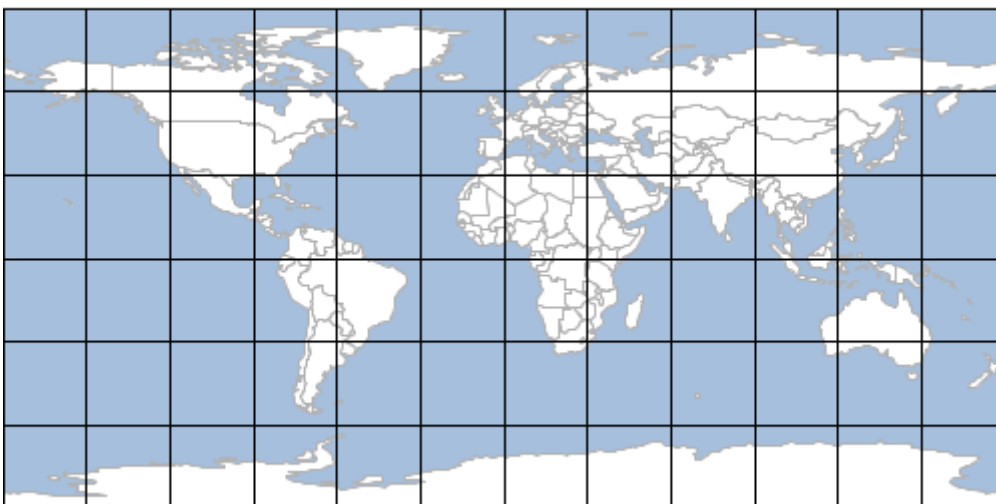
Create a square graticules Layer

```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")
double length = 20
double spacing = 5
Layer layer = Graticule.createSquares(bounds, length, spacing)
```



Create a square graticules Shapefile Layer

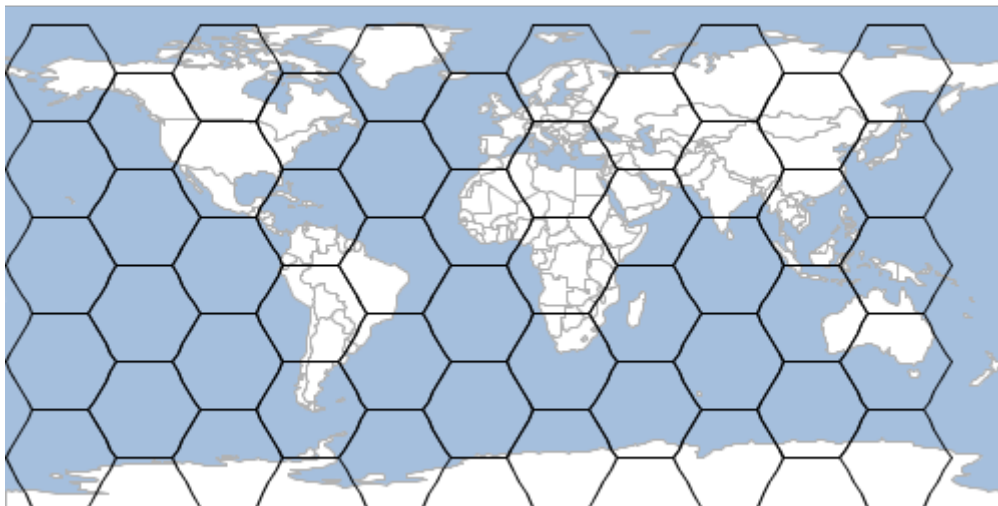
```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")
double length = 30
double spacing = -1
Workspace workspace = new Directory("target")
Layer layer = Graticule.createSquares(bounds, length, spacing, workspace: workspace,
layer: "squares")
```



Hexagon

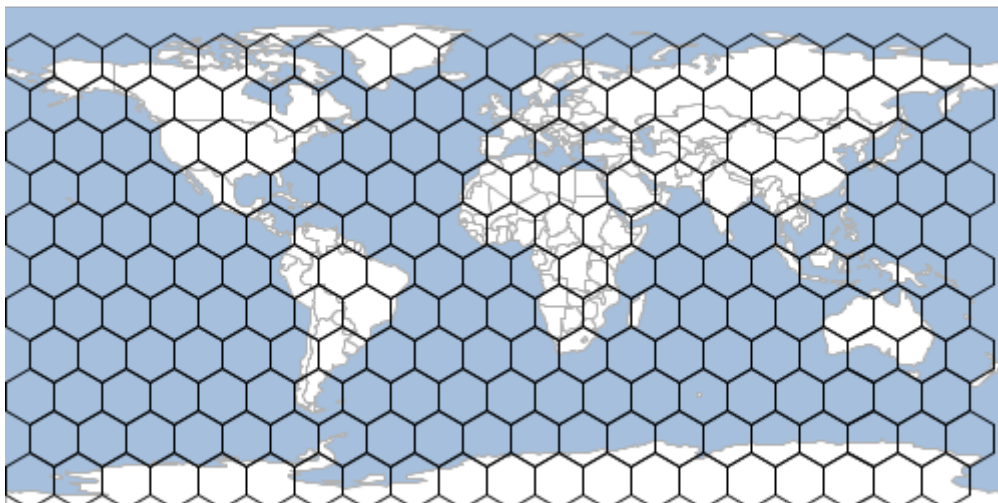
Create a flat hexagon graticules Layer

```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")
double length = 20
double spacing = 5
String orientation = "flat"
Layer layer = Graticule.createHexagons(bounds, length, spacing, orientation)
```



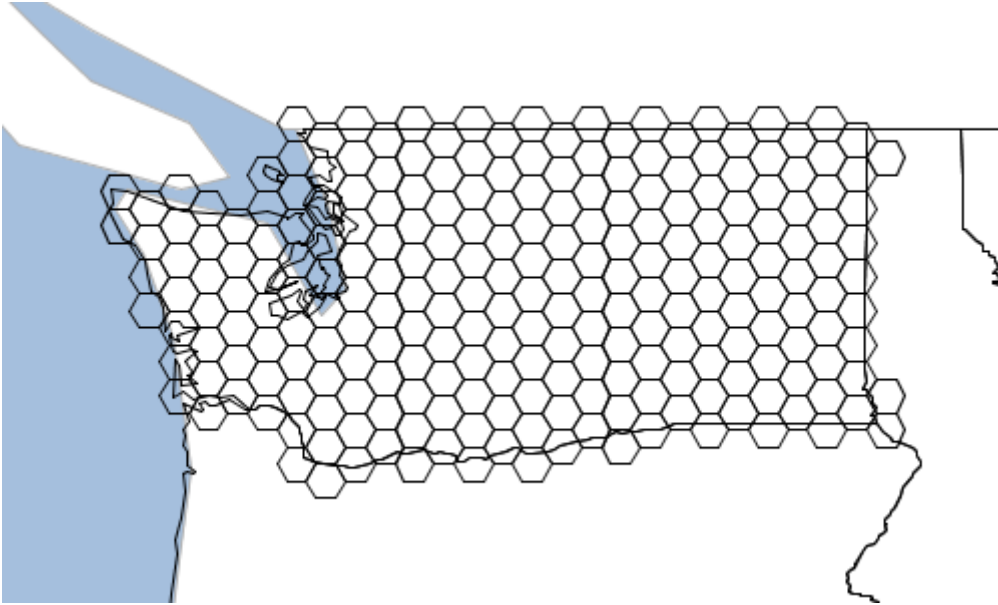
Create a angled hexagon graticules Layer

```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")
double length = 10
double spacing = 5
String orientation = "angled"
Layer layer = Graticule.createHexagons(bounds, length, spacing, orientation)
```



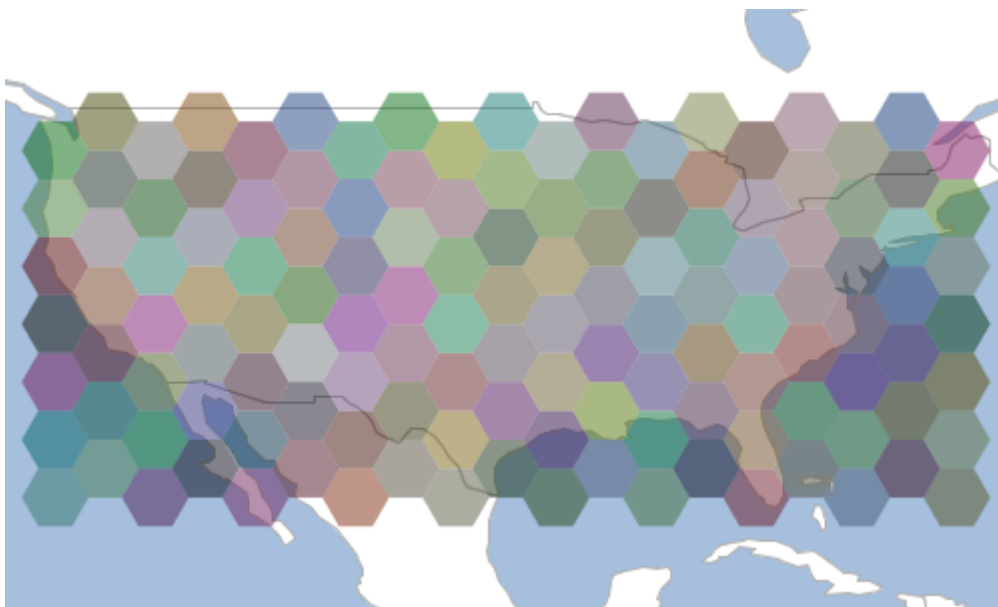
Create a hexagon graticules Layer intersecting Washington States

```
Layer states = new Shapefile("src/main/resources/data/states.shp")
Feature feature = states.first(filter: "STATE_NAME = 'Washington'")
Layer layer = Graticule.createHexagons(feature.bounds.expandBy(1.0), 0.2, -1.0,
"flat", createFeature: { GridElement e ->
    new Point(e.center.x, e.center.y).buffer(0.2).intersects(feature.geom)
})
```



Create a hexagon graticules Layer with a custom schema

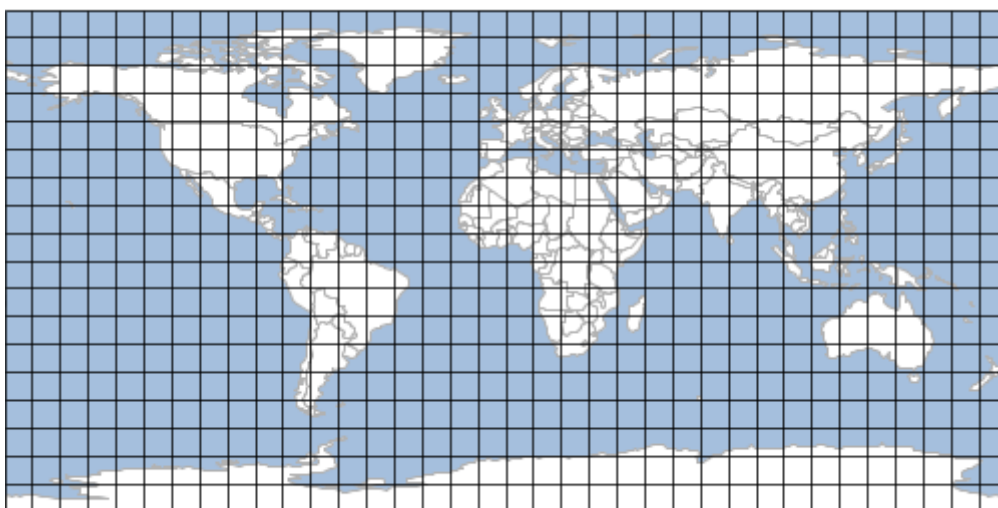
```
Layer states = new Shapefile("src/main/resources/data/states.shp")
Schema schema = new Schema("hexagon", [
    new Field("geom", "Polygon"),
    new Field("color", "java.awt.Color")
])
Bounds b = states.bounds.expandBy(1)
Layer layer = Graticule.createHexagons(b, 2.0, -1.0, "flat", schema: schema,
setAttributes: { GridElement e, Map attributes ->
    attributes["color"] = Color.randomPastel.asColor()
})
layer.style = new Fill(new Property("color"), 0.5)
```



Line

Create a line graticules Layer

```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")
Layer layer = Graticule.createLines(bounds, [
    [orientation: 'vertical', level: 2, spacing: 20],
    [orientation: 'vertical', level: 1, spacing: 10 ],
    [orientation: 'horizontal', level: 2, spacing: 20],
    [orientation: 'horizontal', level: 1, spacing: 10 ]
], 2.0)
```



Rectangle

Create a rectangular graticules Layer

```
Bounds bounds = new Bounds(-180,-90,180,90,"EPSG:4326")  
double width = 20  
double height = 10  
Layer layer = Graticule.createRectangles(bounds, width, height, -1)
```

