

Table of Contents

Render Recipes	1
Creating Maps	1
Map Properties	5
Advanced Properties	8
Map Cubes	12
Rendering Maps	15
Displaying Maps	33
Drawing	35
Plotting	43
Reading Maps	51

Render Recipes

The Render classes are in the [geoscript.render](#) package.

Creating Maps

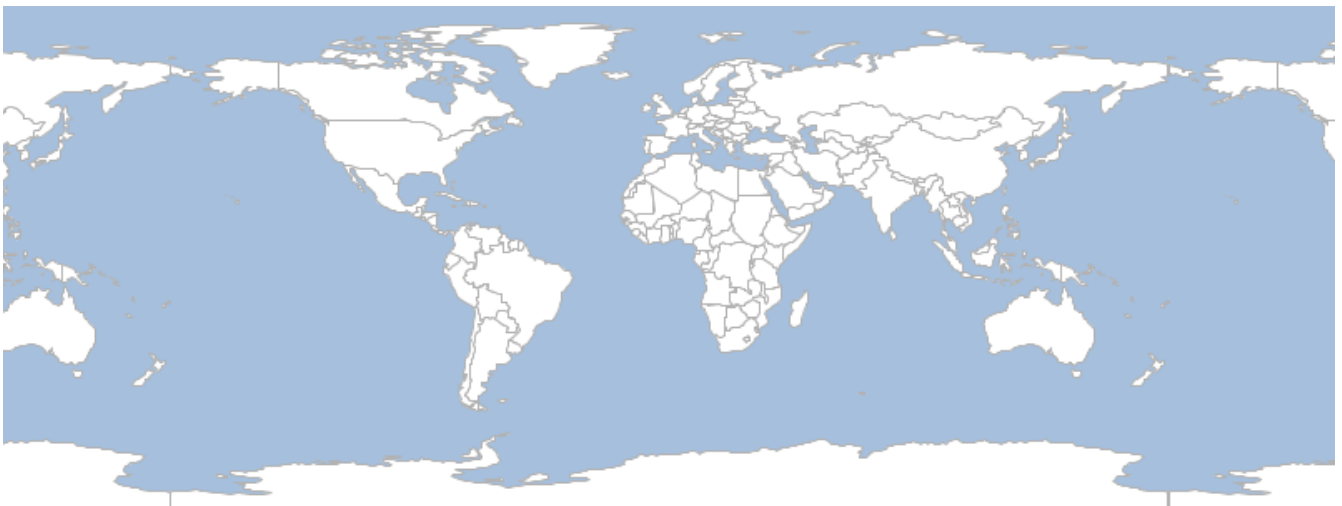
Create a Map with Layers and render to a File.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
File file = new File("map.png")
map.render(file)
```



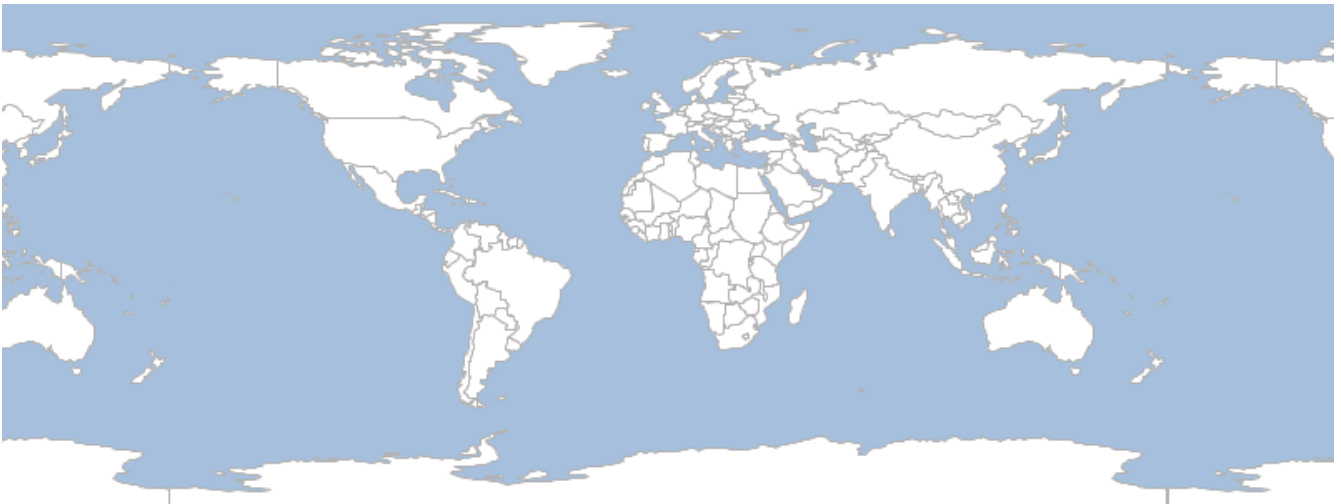
Create a Map with Layers and render to a file name.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfd9")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
map.render("map.png")
```



Create a Map with Layers and render to an OutputStream.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->
    map.render(outputStream)
}
```



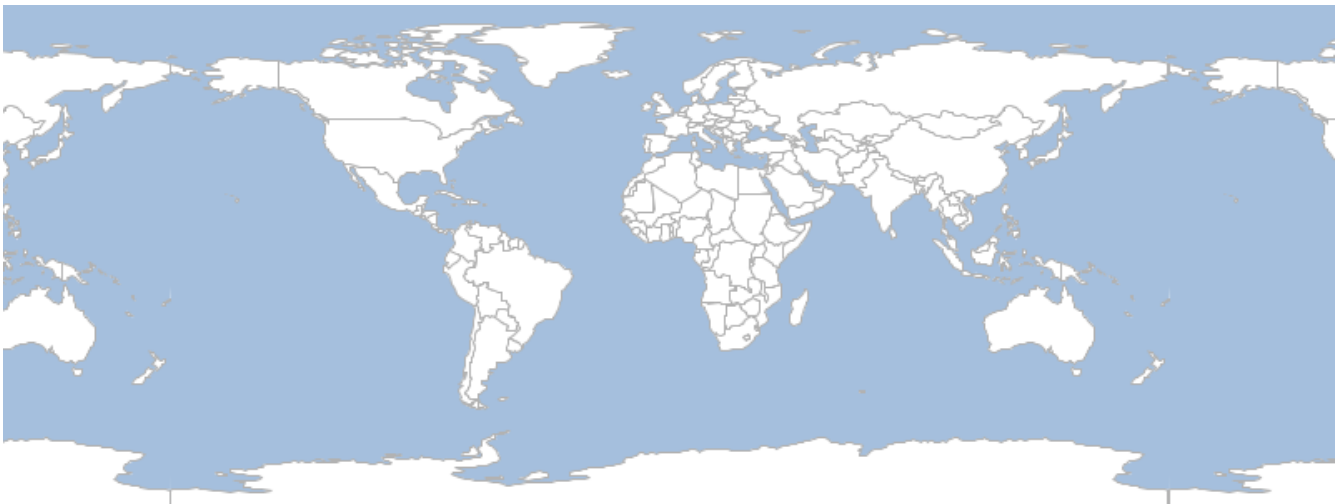
Create a Map with Layers and render to an BufferedImage.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
BufferedImage image = map.renderToImage()
```



Create a Map with Layers and render to a Graphics2D object.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
BufferedImage image = new BufferedImage(800, 300, BufferedImage.TYPE_INT_ARGB)
Graphics2D graphics = image.graphics
map.render(graphics)
graphics.dispose()
```



Create a Map with Layers and display in a simple UI.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfd9")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
map.display()
```



Map Properties

Get Map properties

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Map map = new Map(
    width: 600,
    height: 600,
    backgroundColor: "#a5bfdd",
    layers: [countries],
    type: "png",
    proj: "EPSG:3857",
    bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
    fixAspectRatio: false
)
File file = new File("map.png")
map.render(file)
```



Get width and height

```
int width = map.width  
int height = map.height  
println "Width and Height = ${width} x ${height}"
```

Width and Height = 600 x 600

Get the Bounds

```
Bounds bounds = map.bounds  
println "Bounds = ${bounds}"
```

Bounds = (-2.0037508342789244E7,-
1.9971868880408555E7,2.0037508342789244E7,1.9971868880408563E7,EPG:3857)

Get the Projection

```
Projection projection = map.proj  
println "Projeciton = ${projection}"
```

Projeciton = EPSG:3857

Get the Layers

```
List<Layer> layers = map.layers  
println "Layers:"  
layers.each { Layer layer ->  
    println "    ${layer.name}"  
}
```

Layers:
countries

Get the renderer type

```
String type = map.type  
println "Type = ${type}"
```

Type = png

Get whether we are fixing the aspect ratio or not.

```
boolean shouldFixAspectRatio = map.fixAspectRatio
println "Fix Aspect Ratio = ${shouldFixAspectRatio}"
```

```
Fix Aspect Ratio = false
```

Get the background color

```
String backgroundColor = map.backgroundColor
println "Background Color = ${backgroundColor}"
```

```
Background Color = #a5bfdd
```

Get the scale

```
double scale = map.scaleDenominator
println "Scale = ${scale}"
```

```
Scale = 2.385417659855862E8
```

Advanced Properties

You can set the scale computation to be accurate (the default) or ogc compliant.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 400,
    height: 300,
    layers: [ocean, countries],
    bounds: new Bounds(-162.070313, 9.968851, -35.507813, 58.995311, "EPSG:4326")
)

map.setScaleComputation("accurate")
File accurateFile = new File("map_accurate.png")
map.render(accurateFile)

map.setScaleComputation("ogc")
File ogcFile = new File("map_ogc.png")
map.render(ogcFile)
```


Accurate



OGC



You can set whether to use advanced projection handling or not. By default this is set to true.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 400,
    height: 300,
    layers: [ocean, countries],
    bounds: new Bounds(-162.070313, 9.968851, -35.507813, 58.995311, "EPSG:4326")
)

map.setAdvancedProjectionHandling(true)
File trueFile = new File("map_advancedproj_true.png")
map.render(trueFile)

map.setAdvancedProjectionHandling(false)
File falseFile = new File("map_advancedproj_false.png")
map.render(falseFile)
```

Yes



No



You can set whether to use continuous map wrapping. The default is true.

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 200,
    layers: [ocean, countries]
)

map.setContinuousMapWrapping(true)
File trueFile = new File("map_continuouswrapping_true.png")
map.render(trueFile)

map.setContinuousMapWrapping(false)
File falseFile = new File("map_continuouswrapping_false.png")
map.render(falseFile)
```

Yes



No



Map Cubes

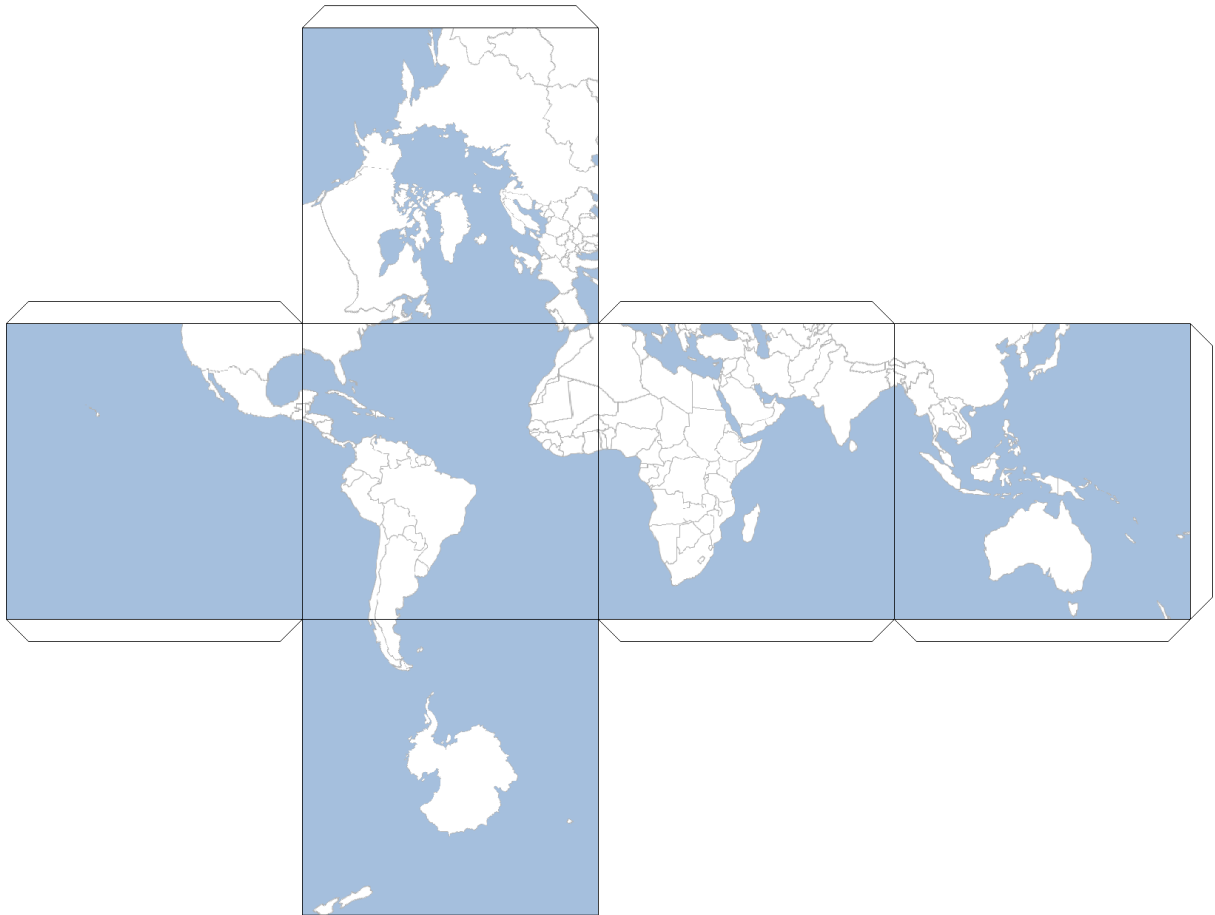
Create a map cube to a file

```
Workspace workspace = new Directory("src/main/resources/shapefiles")
Layer countries = workspace.get("countries")
Layer ocean = workspace.get("ocean")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
ocean.style = new Fill("#a5bfdd")

MapCube mapCube = new MapCube(
    drawOutline: true,
    drawTabs: true,
    tabSize: 30,
    title: "World Cube",
    source: "Natural Earth",
    imageType: "png"
)
File file = new File("map_cube_file.png")
mapCube.render([ocean, countries], file)
```

World Cube

Natural Earth



Create a map cube to a byte array

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")

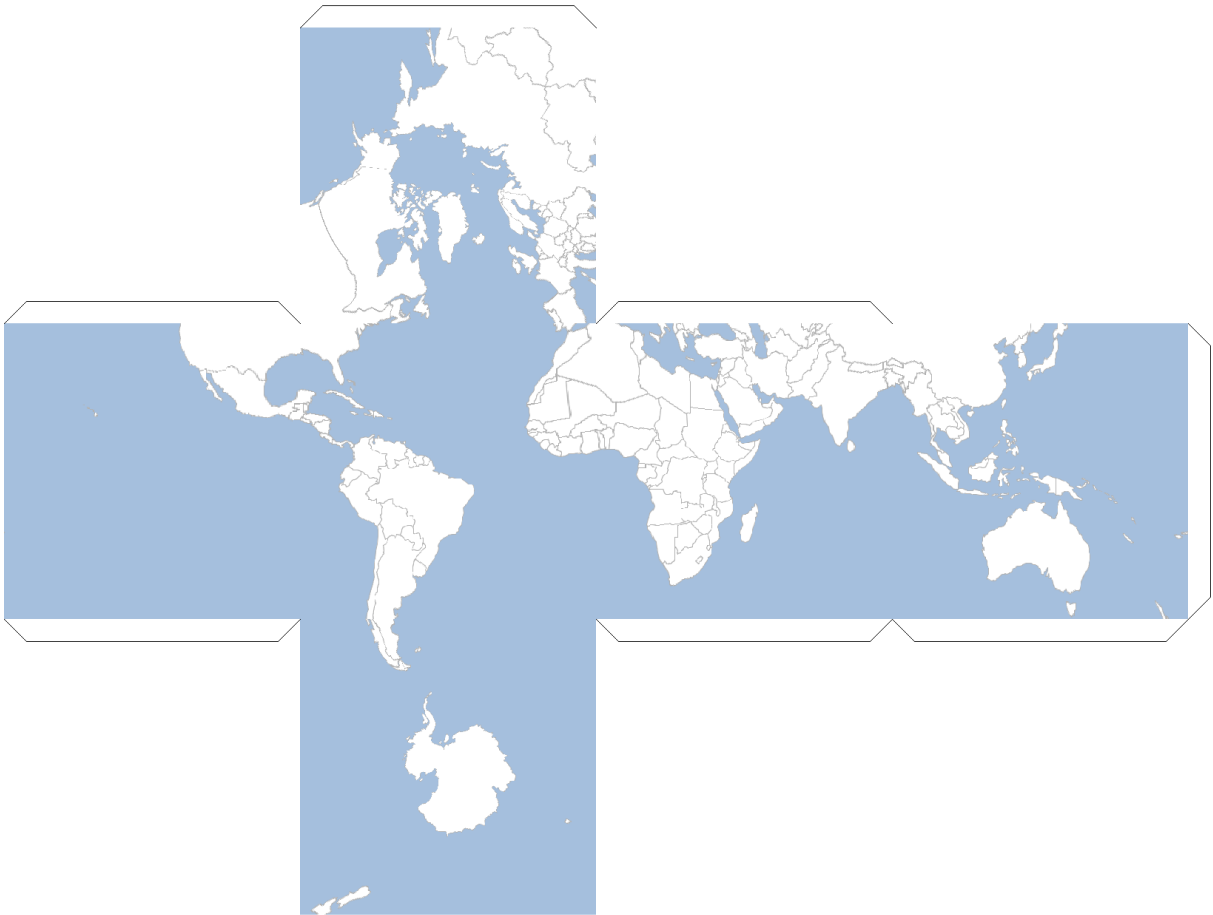
MapCube mapCube = new MapCube()
byte[] bytes = mapCube.render([ocean, countries])
```



Create a map cube to a byte array

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")

MapCube mapCube = new MapCube()
File file = new File("map_cube_stream.png")
file.withOutputStream { OutputStream outputStream ->
    mapCube.render([ocean, countries], outputStream)
}
```



Rendering Maps

Finding Renderers

Get all Renderers

```
List<Renderer> renderers = Renderers.list()
renderers.each { Renderer renderer ->
    println renderer.class.simpleName
}
```

ASCII
Base64
GeoTIFF
GIF
JPEG
Pdf
PNG
Svg

Get a Renderer

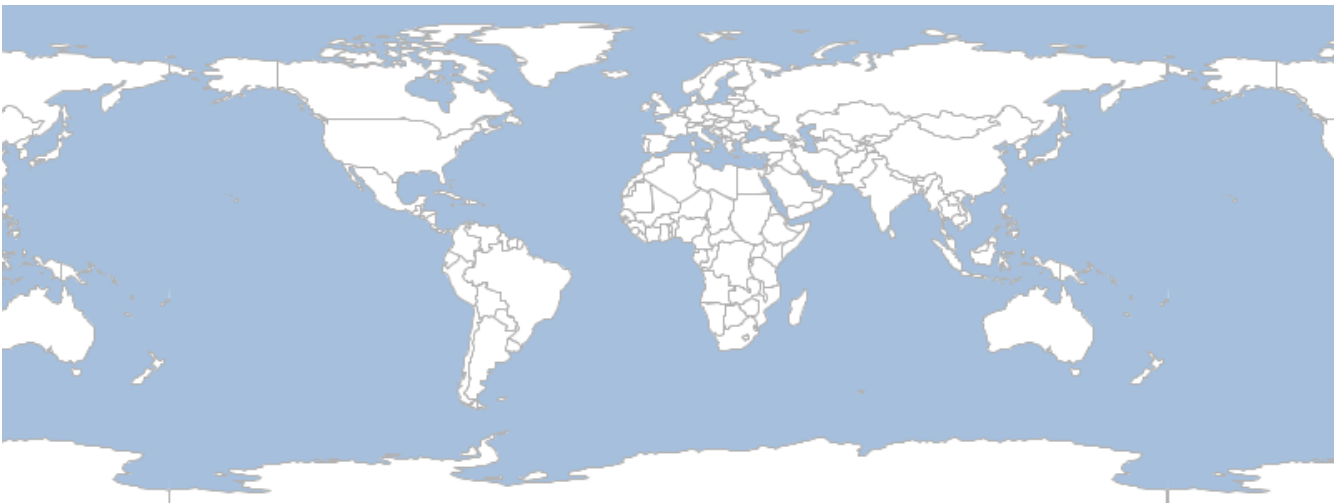
```
Renderer renderer = Renderers.find("png")
println renderer.class.simpleName
```

PNG

Image

Render a Map to an image using an Image Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Image png = new Image("png")
BufferedImage image = png.render(map)
```



Render a Map to an OutputStream using the Image Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Image jpeg = new Image("jpeg")
File file = new File("map.jpeg")
FileOutputStream out = new FileOutputStream(file)
jpeg.render(map, out)
out.close()
```



PNG

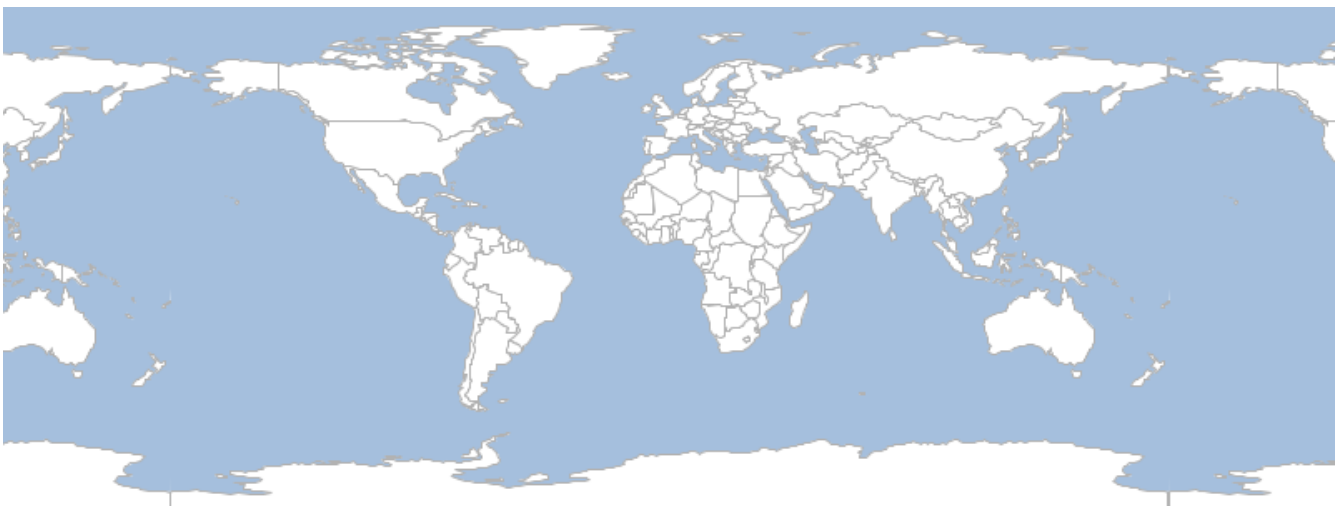
Render a Map to an Image using the PNG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
PNG png = new PNG()
BufferedImage image = png.render(map)
```



Render a Map to an OutputStream using the PNG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
PNG png = new PNG()
File file = new File("map.png")
FileOutputStream out = new FileOutputStream(file)
png.render(map, out)
out.close()
```



JPEG

Render a Map to an Image using the JPEG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
JPEG jpeg = new JPEG()
BufferedImage image = jpeg.render(map)
```



Render a Map to an OutputStream using the JPEG Renderer

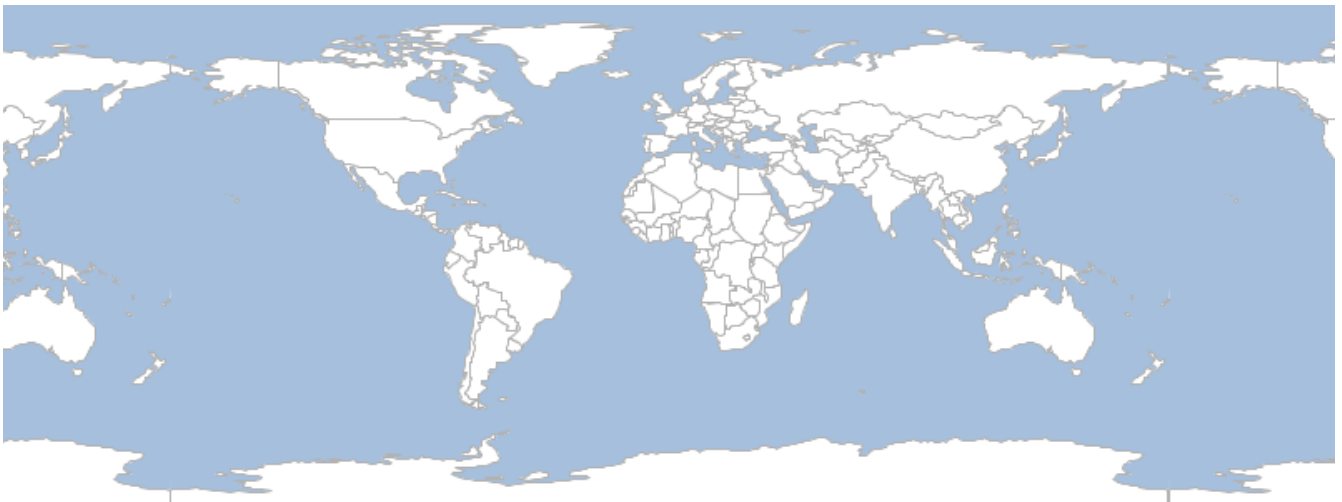
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
JPEG jpeg = new JPEG()
File file = new File("map.jpeg")
FileOutputStream out = new FileOutputStream(file)
jpeg.render(map, out)
out.close()
```



GIF

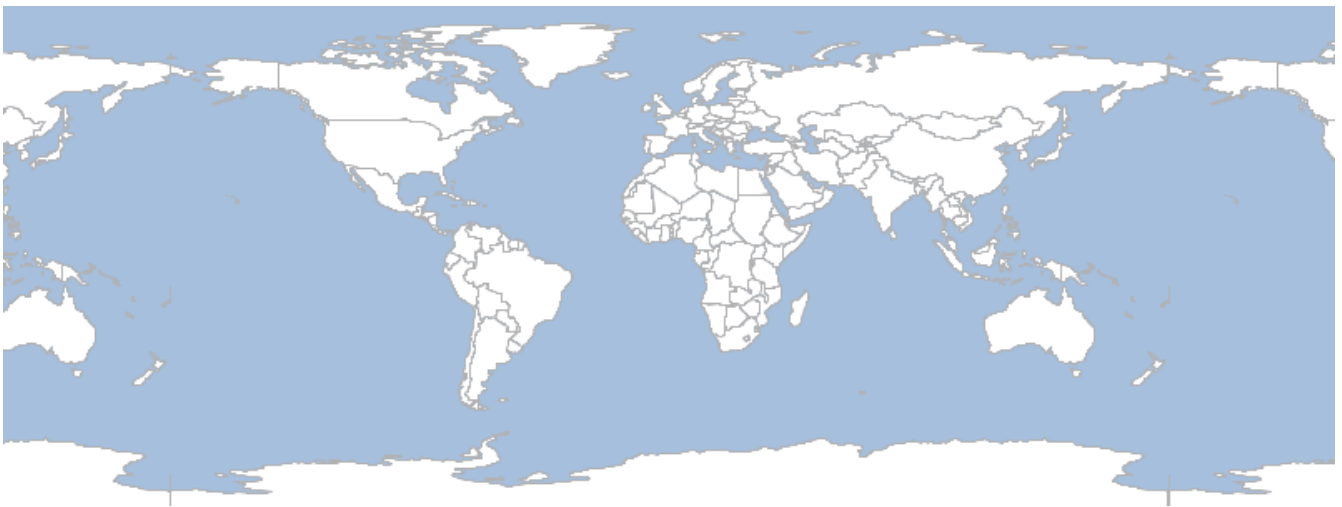
Render a Map to an Image using the GIF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfd9")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
GIF gif = new GIF()
BufferedImage image = gif.render(map)
```



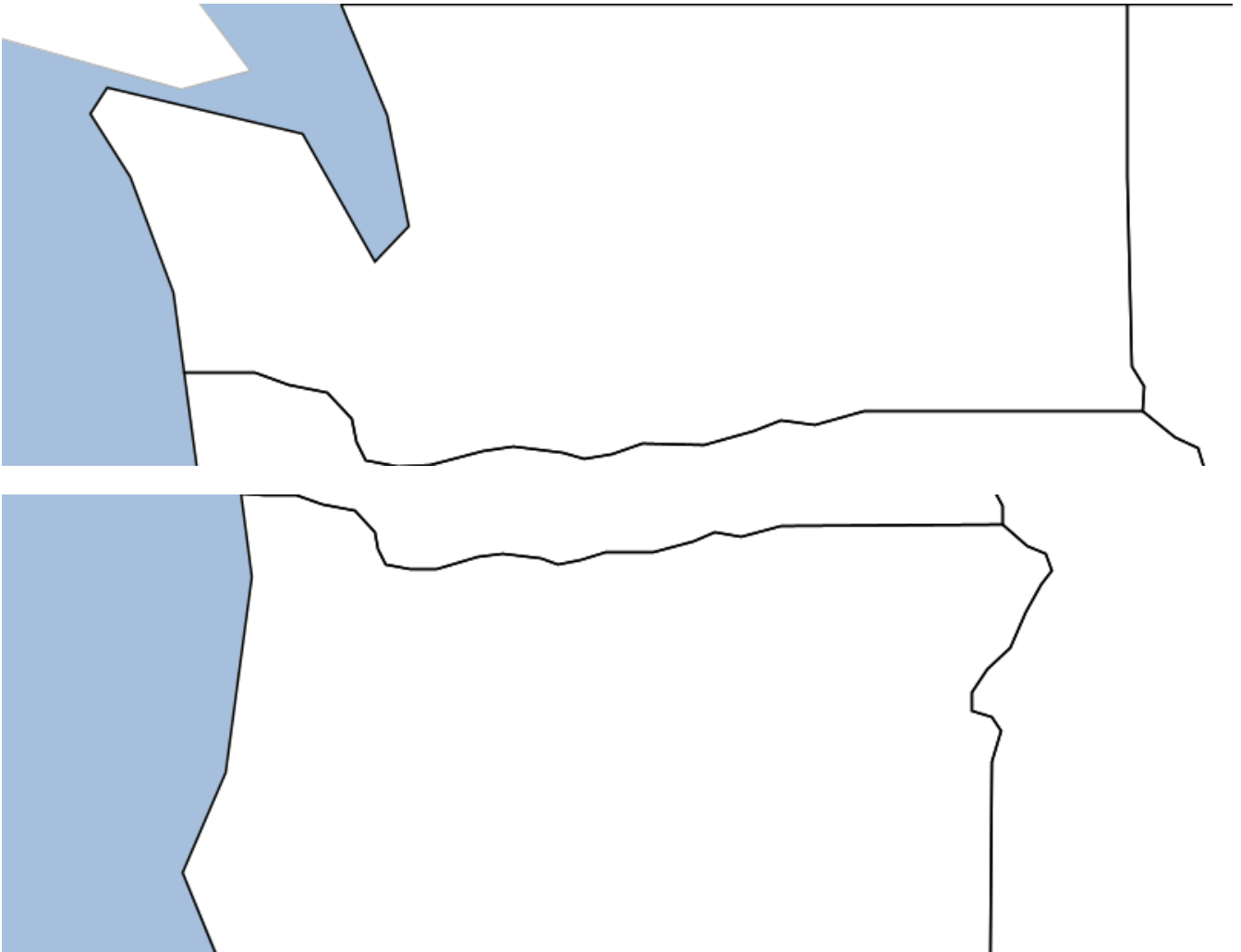
Render a Map to an OutputStream using the GIF Renderer

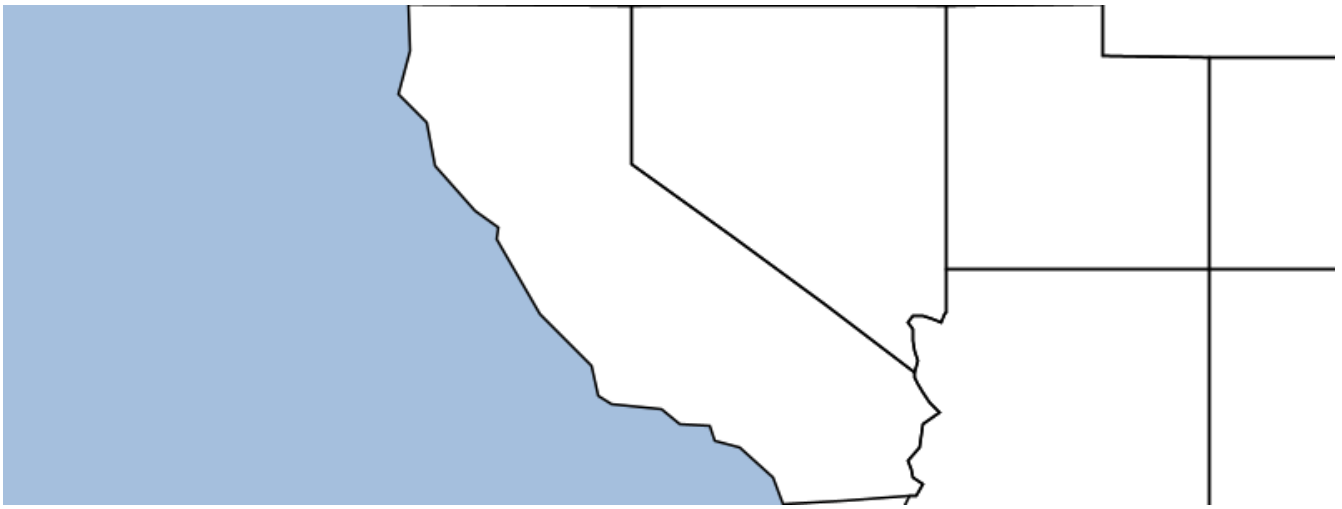
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
GIF gif = new GIF()
File file = new File("map.gif")
gif.render(map, new FileOutputStream(file))
```



```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer states = workspace.get("states")
states.style = new Fill("") + new Stroke("black", 1.0)
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries, states]
)

GIF gif = new GIF()
List images = ["Washington", "Oregon", "California"].collect { String state ->
    map.bounds = states.getFeatures("NAME_1 = '${state}')[0].bounds
    def image = gif.render(map)
    image
}
File file = new File("states.gif")
gif.renderAnimated(images, file, 500, true)
```

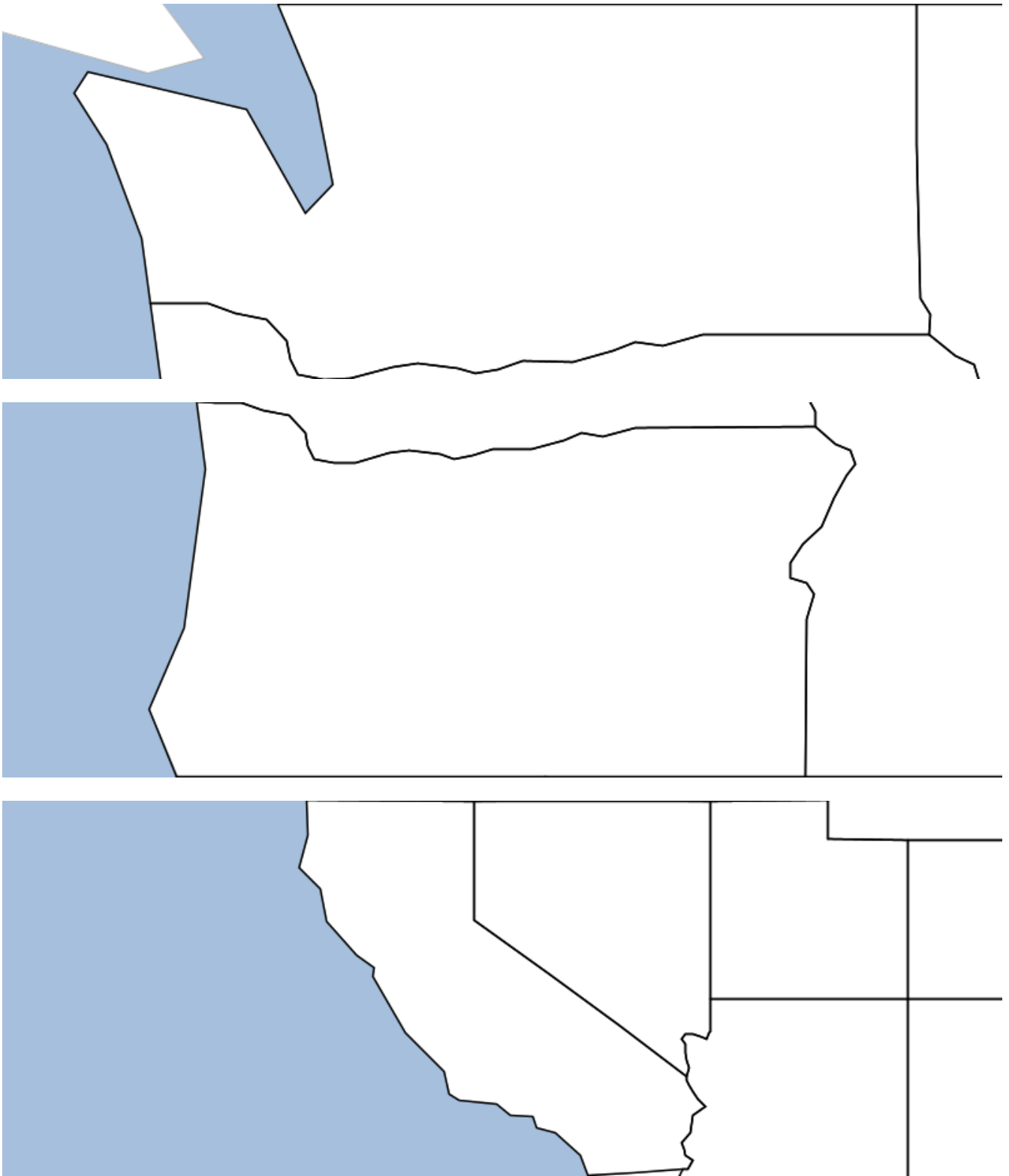




Render a Map to an animated GIF to a byte array using the GIF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer states = workspace.get("states")
states.style = new Fill("") + new Stroke("black", 1.0)
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries, states]
)

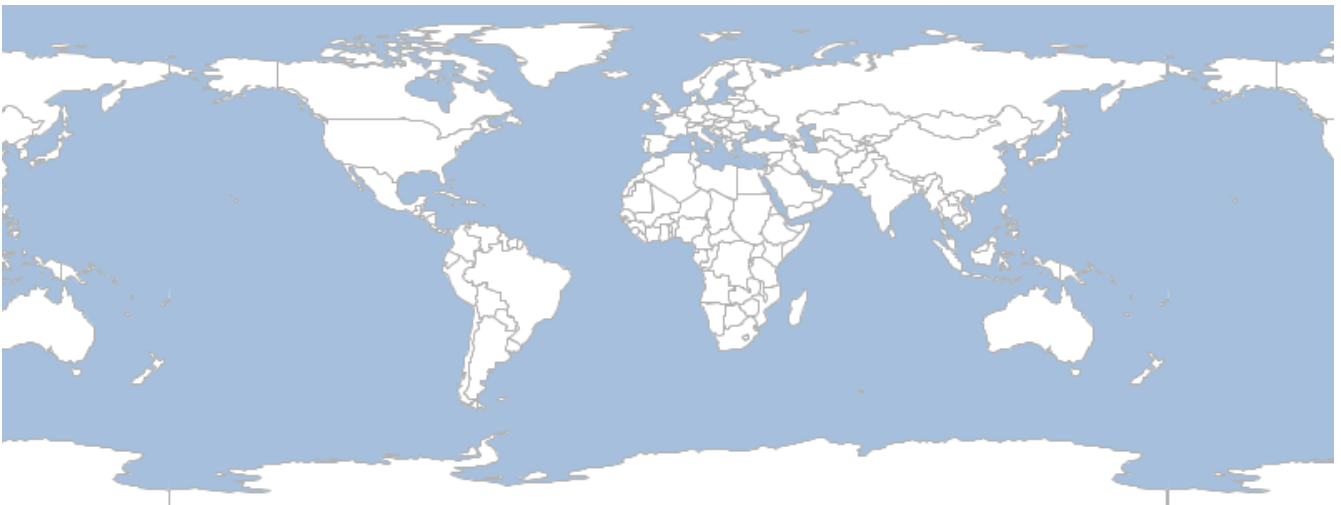
GIF gif = new GIF()
List images = ["Washington", "Oregon", "California"].collect { String state ->
    map.bounds = states.getFeatures("NAME_1 = '${state}')[0].bounds
    def image = gif.render(map)
    image
}
File file = new File("states.gif")
byte[] bytes = gif.renderAnimated(images, 500, true)
file.bytes = bytes
```



GeoTIFF

Render a Map to an Image using the GeoTIFF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
GeoTIFF geotiff = new GeoTIFF()
BufferedImage image = geotiff.render(map)
```



Render a Map to an OutputStream using the GeoTIFF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
GeoTIFF geotiff = new GeoTIFF()
File file = new File("map.tif")
geotiff.render(map, new FileOutputStream(file))
```



ASCII

Render a Map to an string using the ASCII Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
ASCII ascii = new ASCII(width: 60)
String asciiStr = ascii.render(map)
println asciiStr
```

```

((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
.....:^(^..!?!^!^:..!(^...(((((^.^(((((.....:^(^..!?!^
.....(:((.....((.(.((.(.(((((:(.....:((.....
?..^((((((((^.....((((((.(.....?..^((((((((^
.^(((((((((((^.....(((((((...:^^?...(...:^((((((((((((^
((.((((((((((((.....:(((((((^(((!^((^.....((.((((((((((((
(((((((((((((((((((.:(((((((((((.....:(((((((((((((((
(((((((((((^(((((((.(.(((((((.....:.....((..^((((((((((((^
(((((((((((((((((((.(.(((((((^:(...^(((.(^(((((((((((((((
((((((((((((((((((((!:(((((((.....(((((((((((((((((((
((.((((((((((((((((.....(((((((...(((((((((((.(.(((((((((((
(^(((((((((((((((((((.....(((((((.^(((((((((((^(((((((((((
....(((((((((((((((((((...(((((((.(^(((((((((((^.....(((((((((((
....(((((((((((((((((((...(((((((((((:((((((((((((.....(((((((((((
(((.((((((((((((((((((((.(.((((((((((((((((((((((((((((((((
(((((((((((((((((((((((.(.((((((((((((((((((((((((((((((((
(((((((((((((((((((((((^((((((((((((((((((((((((((((((((
((((((((((((((((((((((((((((((((((((((((((((((((((((((((
....(((((((((((((((((((?(((((((((((?....(.....:(((((((((((
.....((((.....((((.....:((((.....
.....^.....

```

Render a Map to an text file using the ASCII Renderer

```

Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
ASCII ascii = new ASCII(width: 60)
File file = new File("map.txt")
FileOutputStream out = new FileOutputStream(file)
ascii.render(map, out)
out.close()

```

```

((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
.....:^(^..!?!^!^:..!(^...(((((^.^(((.....:^(^..!?!^
.....(:((.....((.(.((.(.(((:(.....:((.....
?..^(((((((^(^.....(((((.((.....?..^(((((((^(^
.^((((((((((((^(^.....((((((((.....:^^?...(..:^(((((((((((((
((.(((((((((((((.....:((((((((^(^((:!!^(^.....((.(((((((((((((
((((((((((((((((((.:((((((((((((.....((((((((((((((((
((((((((((((^(^((((((.(.((((((((.....:.....((..^(((((((((((((^(^
((((((((((((((((((((((.(.((((((((^(^:(...^(^((.(^(^((((((((((((
((((((((((((((((((((((!:(((((((((.....((((((((((((((((((((
((.(((((((((((((((((.....((((((((.....((((((((((((((((((((
^(^((((((((((((((((.....((((((((^(^((((((((((((((((
.....((((((((((((((((.....((((((((^(^((((((((^(^.....((((((((
.....((((((((((((((((.....((((((((:(((((((((.....((((((((
((.(((((((((((((((((.....((((((((((((((((((((((((((((((((
((((((((((((((((((((((.(.((((((((((((((((((((((((((((
((((((((((((((((((((((((^(^((((((((((((((((((((((((((((
((((((((((((((((((((((((((((((((((((((((((((((((((((
.....((((((((((((((((((((?((((((((((((?.....((((((((
.....((((.....((((.....((((.....
.....^.....

```

Base64

Render a Map to an string using the Base64 Renderer

```

Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Base64 base64 = new Base64()
String base64Str = base64.render(map)
println base64Str

```

```
image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAYAAAAEsC...
```

Render a Map to an text file using the Base64 Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Base64 base64 = new Base64()
File file = new File("map.txt")
base64.render(map, new FileOutputStream(file))
```

```
iVBORw0KGgoAAAANSUhEUgAAAYAAAABEsCAYAAAA7Ldc6AACAAE...
```

PDF

Render a Map to a PDF Document using the PDF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Pdf pdf = new Pdf()
com.lowagie.text.Document document = pdf.render(map)
```



Render a Map to a PDF file using the PDF Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Pdf pdf = new Pdf()
File file = new File("map.pdf")
pdf.render(map, new FileOutputStream(file))
```



SVG

Render a Map to a SVG Document using the SVG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Svg svg = new Svg()
org.w3c.dom.Document document = svg.render(map)
```



Render a Map to a SVG file using the SVG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfff")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Svg svg = new Svg()
File file = new File("map.svg")
FileOutputStream out = new FileOutputStream(file)
svg.render(map, out)
out.close()
```




Displaying Maps

Finding Displayers

Get all Displayers

```
List<Displayer> displayers = Displayers.list()
displayers.each { Displayer displayer ->
    println displayer.class.simpleName
}
```

MapWindow
Window

Get a Displayer

```
Displayer displayer = Displayers.find("window")
println displayer.class.simpleName
```

Window

Window

Display a Map in a simple GUI

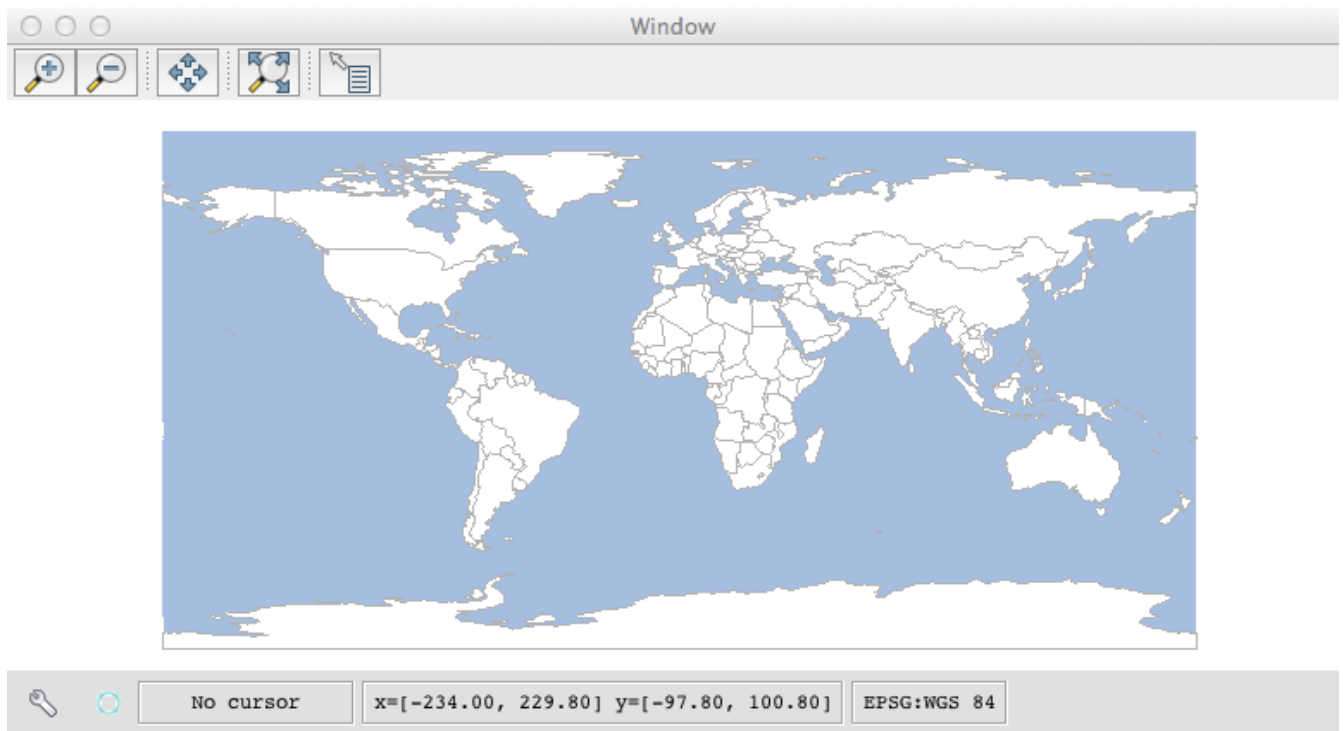
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Window window = new Window()
window.display(map)
```



MapWindow

Display a Map in a interactive GUI

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
MapWindow window = new MapWindow()
window.display(map)
```



Drawing

The Draw class is an easy way to quickly render a Geometry, a List of Geometries, a Feature, or a Layer to an Image, a File, an OutputStream, or a GUI.

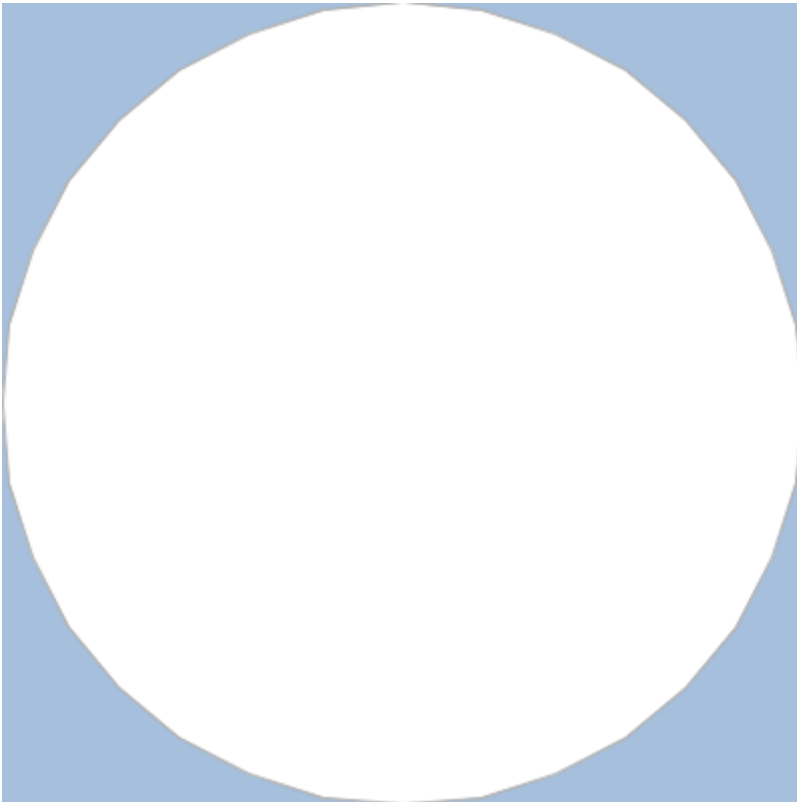
Drawing to a File or GUI

All of the draw methods take a single required parameter but can also take the following optional parameters:

- style = A Style
- bounds = The Bounds
- size = The size of the canvas ([400,350])
- out = The OutputStream, File, or File name. If null (which is the default) a GUI will be opened.
- format = The format ("jpeg", "png", "gif")
- proj = The Projection

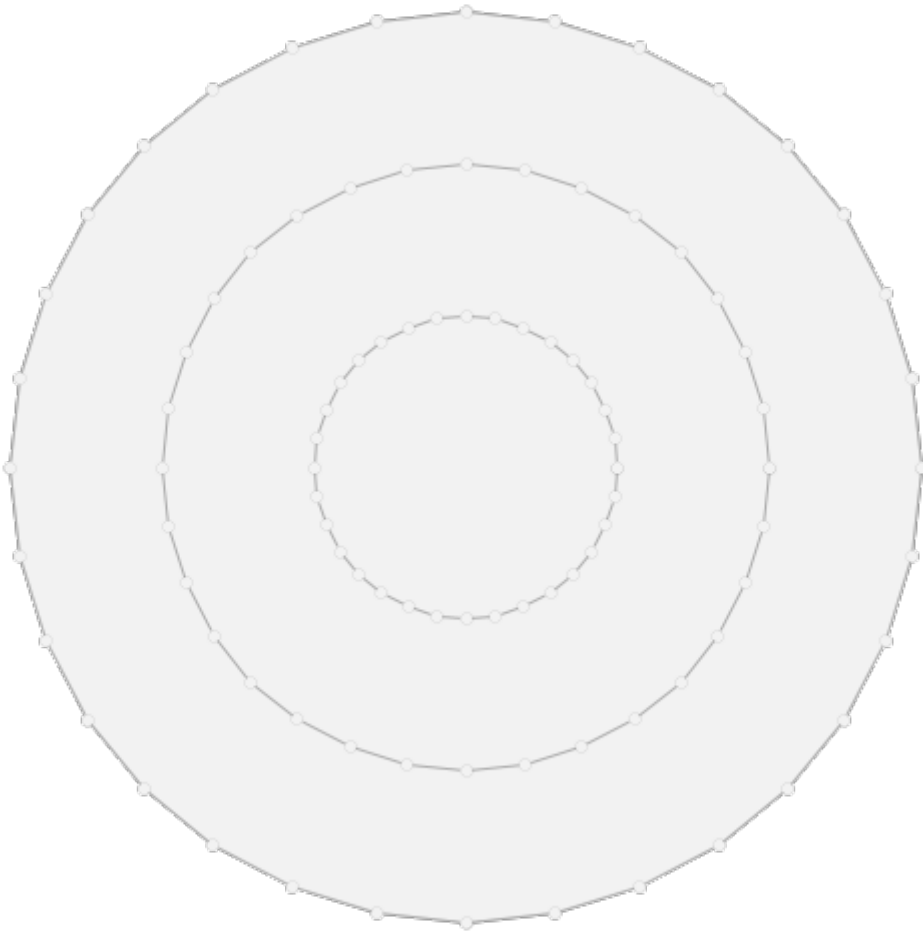
Draw a Geometry to a File

```
File file = new File("geometry.png")
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
Draw.draw(geometry,
    style: new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5),
    bounds: new Bounds(-122.876, 47.087, -121.876, 48.087),
    size: [400, 400],
    format: "png",
    proj: "EPSG:4326",
    backgroundColor: "#a5bfdd",
    out: file
)
```



Draw a List of Geometries to an OutputStream

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
File file = new File("geometries.png")
OutputStream outputStream = new FileOutputStream(file)
Draw.draw(geometries, out: outputStream, format: "png")
outputStream.flush()
outputStream.close()
```



Draw a Feature to a file name

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



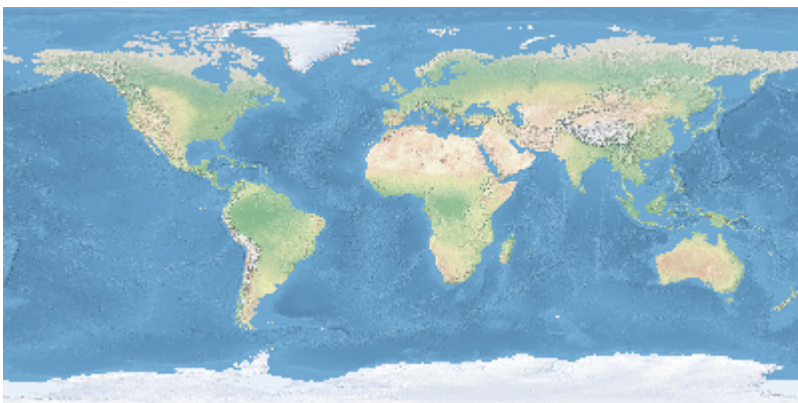
Draw a Layer to a GUI

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
layer.style = new Fill("LightSteelBlue") + new Stroke("LightSlateGrey", 0.25)
Draw.draw(layer, bounds: layer.bounds)
```



Draw a Raster to a File

```
File file = new File("earth.png")
Raster raster = new geoscript.layer.GeoTIFF(new File('src/main/resources/earth.tif'
)).read()
Draw.draw(raster, bounds: raster.bounds, size: [400,200], out: file)
```



Drawing to an Image

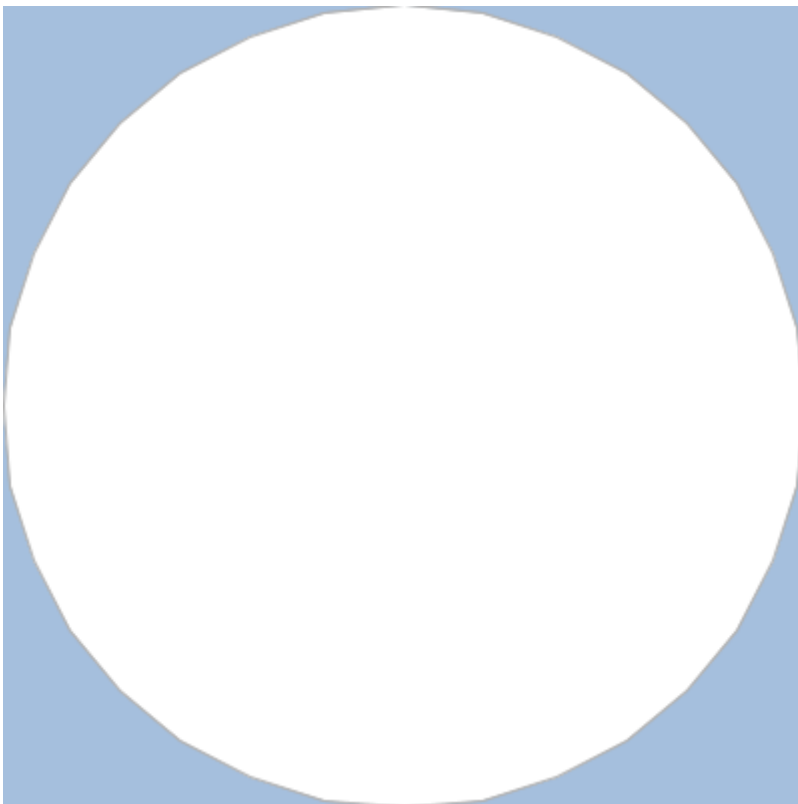
All of the drawToImage methods take a single required parameter but can also take the following optional parameters:

- style = A Style

- bounds = The Bounds
- size = The size of the canvas ([400,350])
- imageType = The format ("jpeg", "png", "gif")
- proj = The Projection

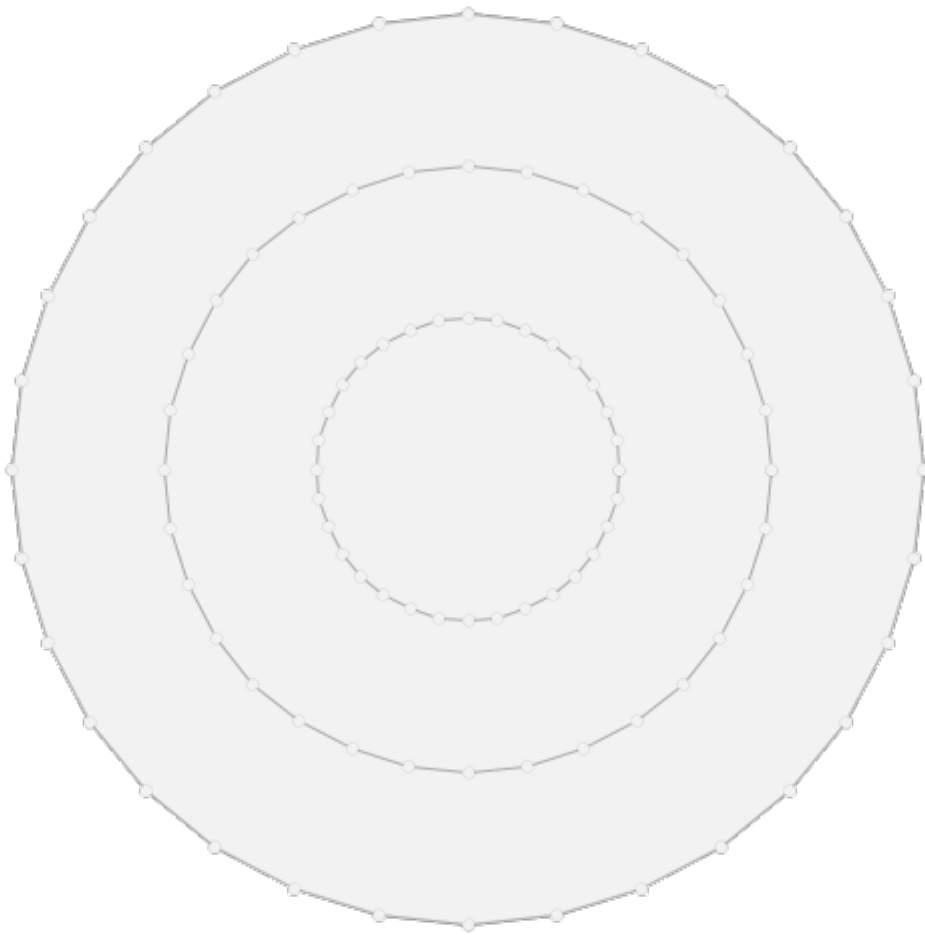
Draw a Geometry to an Image

```
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
BufferedImage image = Draw.drawToImage(geometry,
    style: new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5),
    bounds: new Bounds(-122.876,47.087,-121.876,48.087),
    size: [400,400],
    imageType: "png",
    proj: "EPSG:4326",
    backgroundColor: "#a5bfd9"
)
```



Draw a List of Geometries to an Image

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
BufferedImage image = Draw.drawToImage(geometries)
```

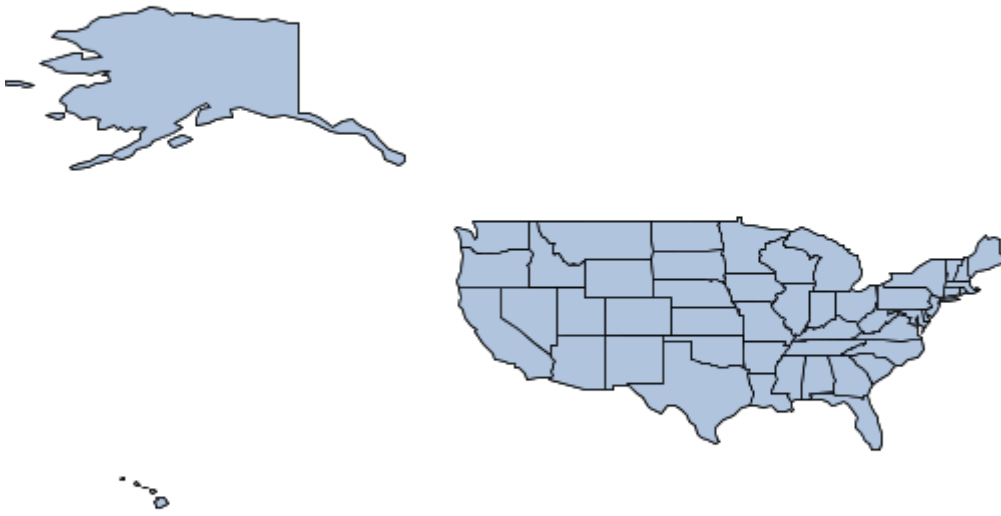
Draw a Feature to an Image

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



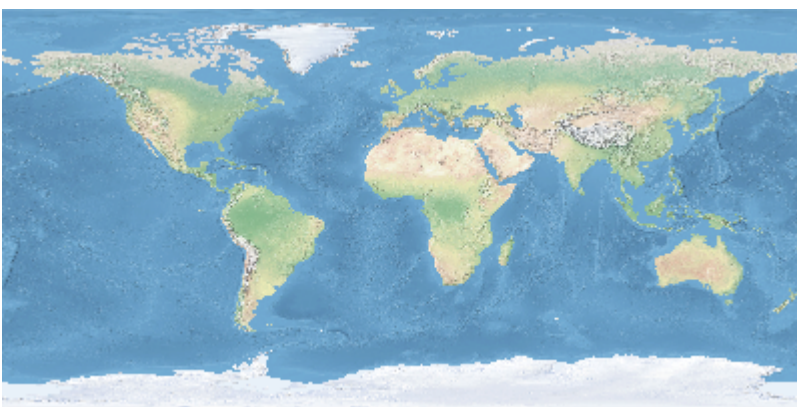
Draw a Layer to an Image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
layer.style = new Fill("LightSteelBlue") + new Stroke("LightSlateGrey", 0.25)
BufferedImage image = Draw.drawToImage(layer, bounds: layer.bounds)
```



Draw a Raster to an Image

```
Raster raster = new geoscript.layer.GeoTIFF(new File('src/main/resources/earth.tif')
).read()
BufferedImage image = Draw.drawImage(raster, bounds: raster.bounds, size: [400,200])
```



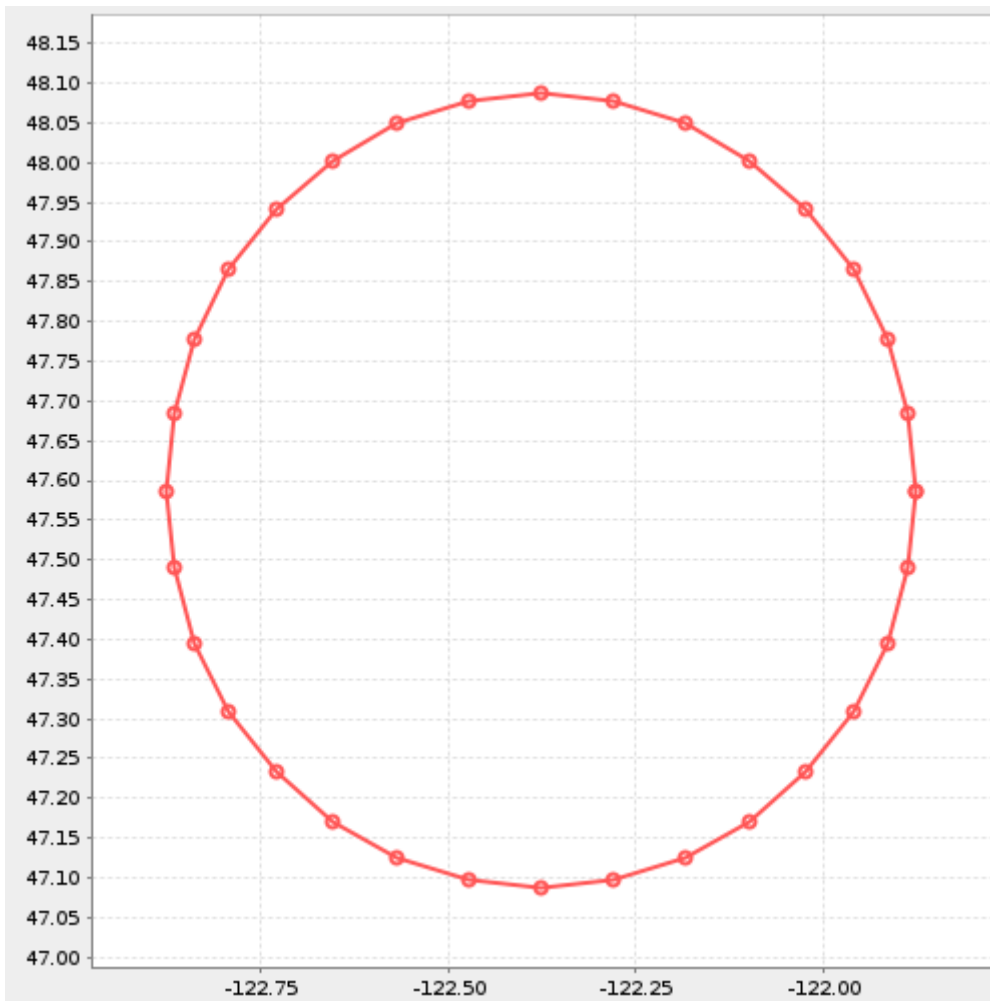
Plotting

Plotting to a File or GUI

The Plot module can plot a Geometry, a list of Geometries, a Feature, or a Layer to a File, a File name, an OutputStream, or a simple GUI.

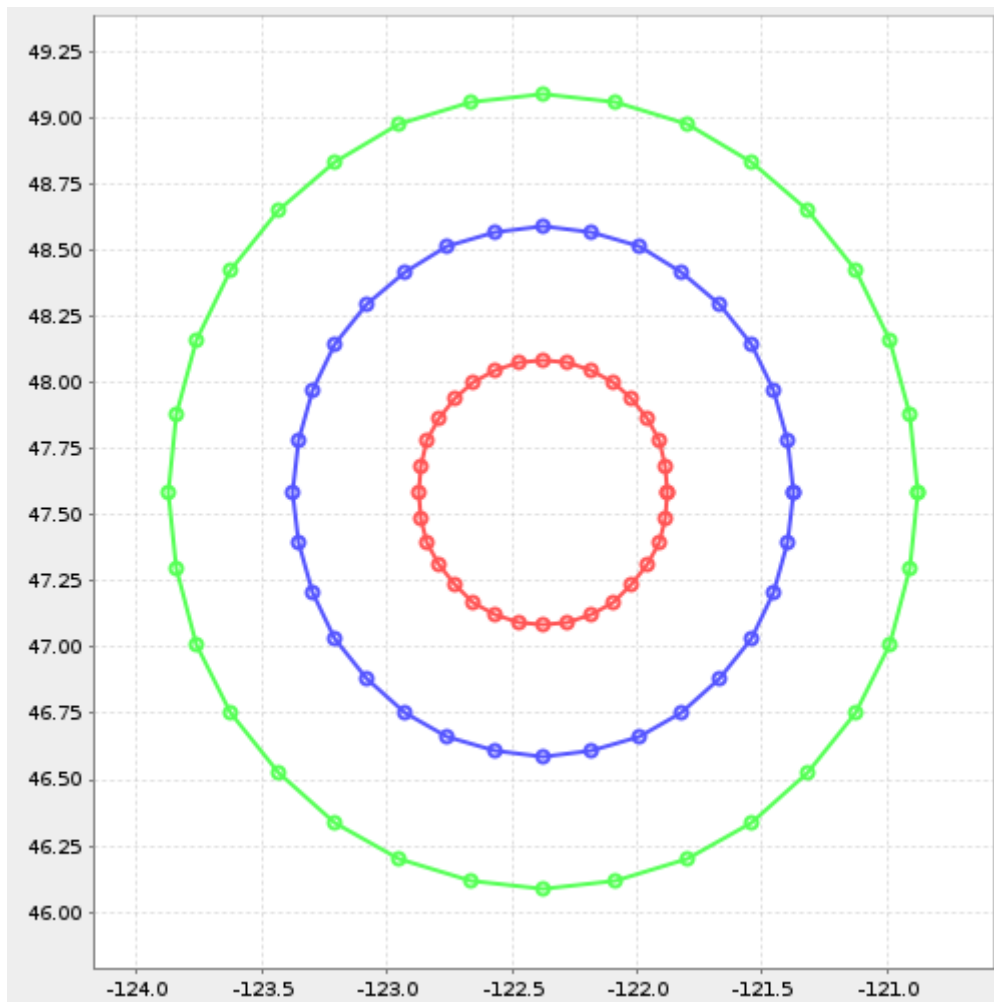
Plot a Geometry to a File

```
File file = new File("geometry.png")
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
Plot.plot(geometry, out: file)
```



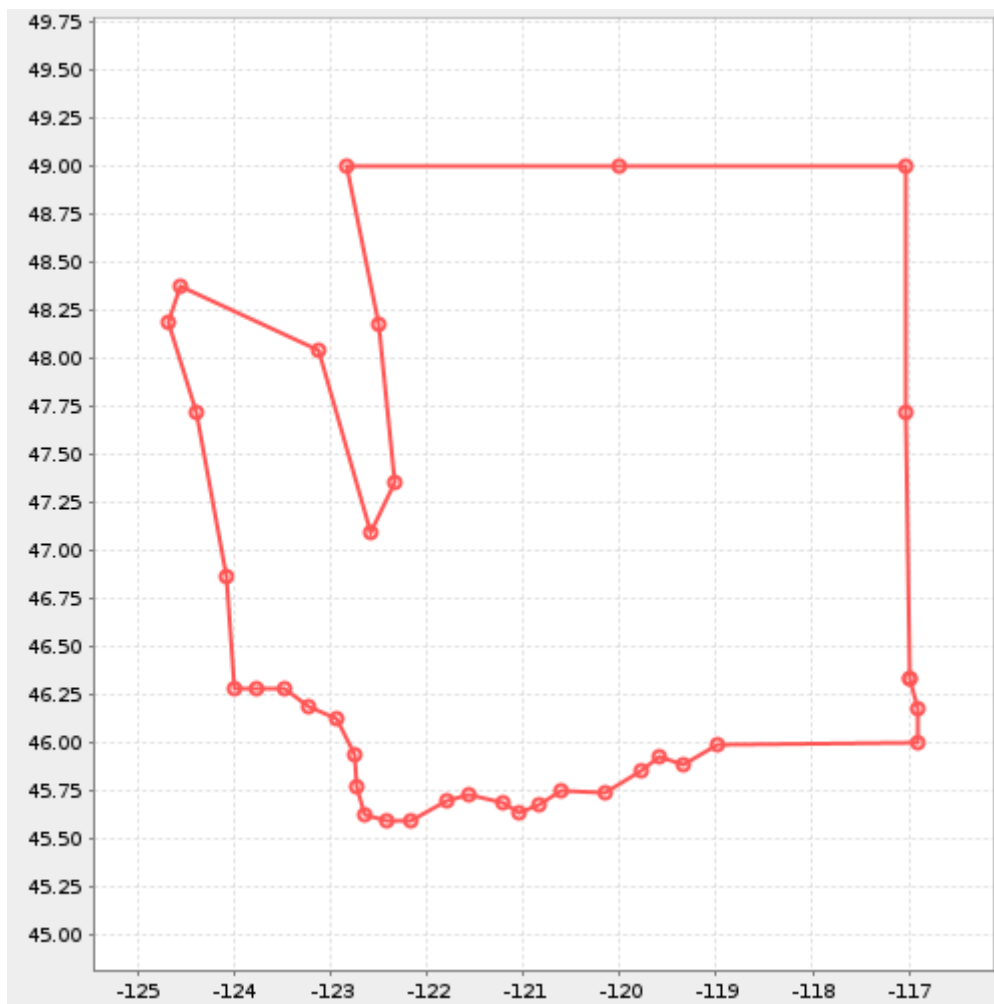
Plot a List of Geometries to an OutputStream

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
File file = new File("geometries.png")
OutputStream outputStream = new FileOutputStream(file)
Plot.plot(geometries, out: outputStream)
outputStream.flush()
outputStream.close()
```



Plot a Feature to a File name

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



Plot a Layer to a GUI

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

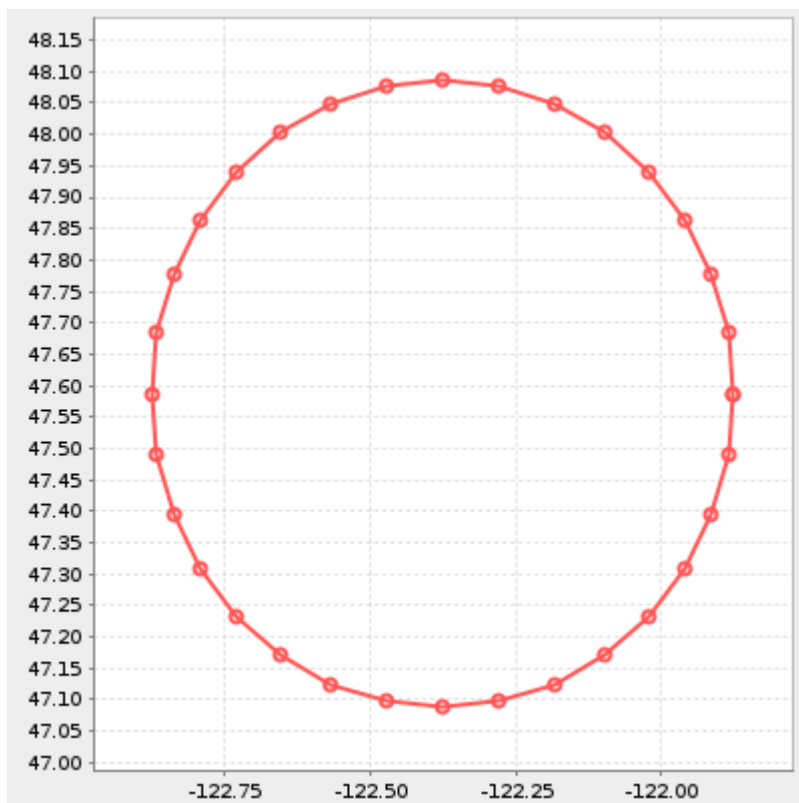


Plotting to an Image

The Plot module can plot a Geometry, a list of Geometries, a Feature, or a Layer to an image.

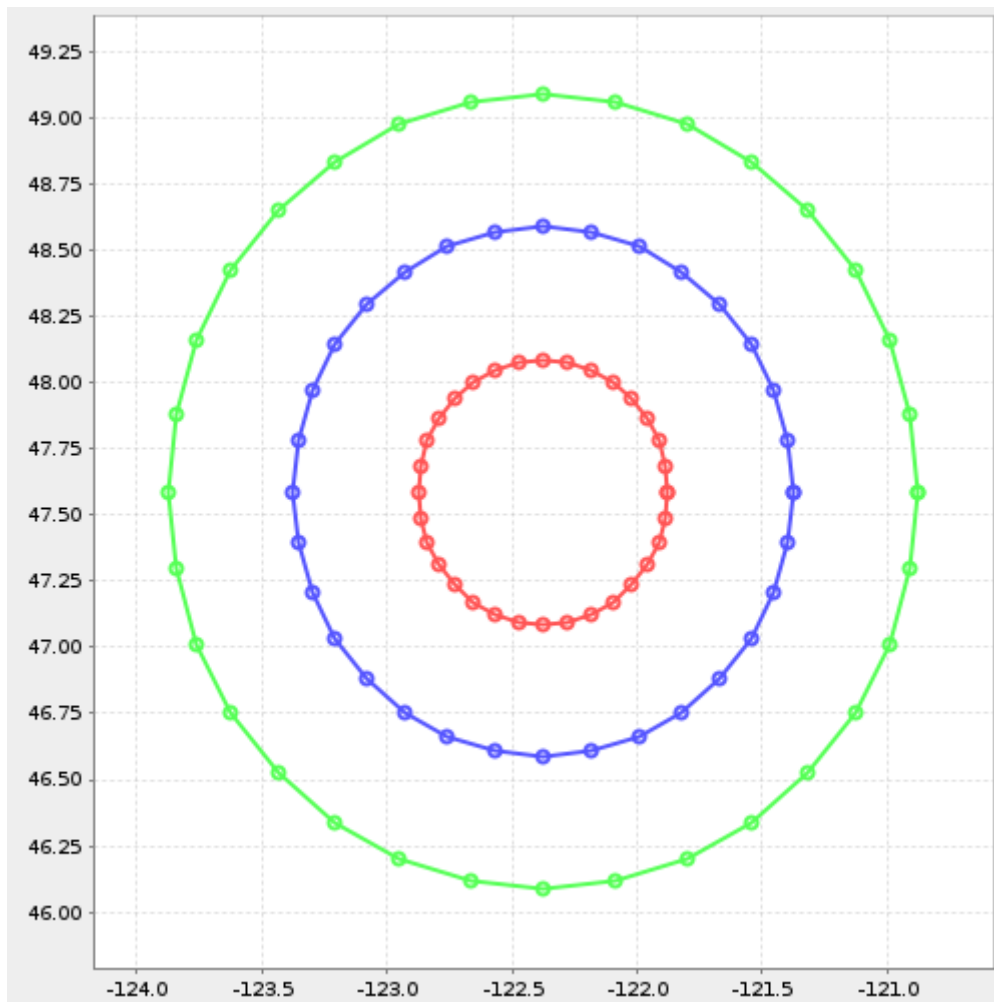
Plot a Geometry to an Image

```
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
BufferedImage image = Plot.plotToImage(geometry, size: [400,400],)
```



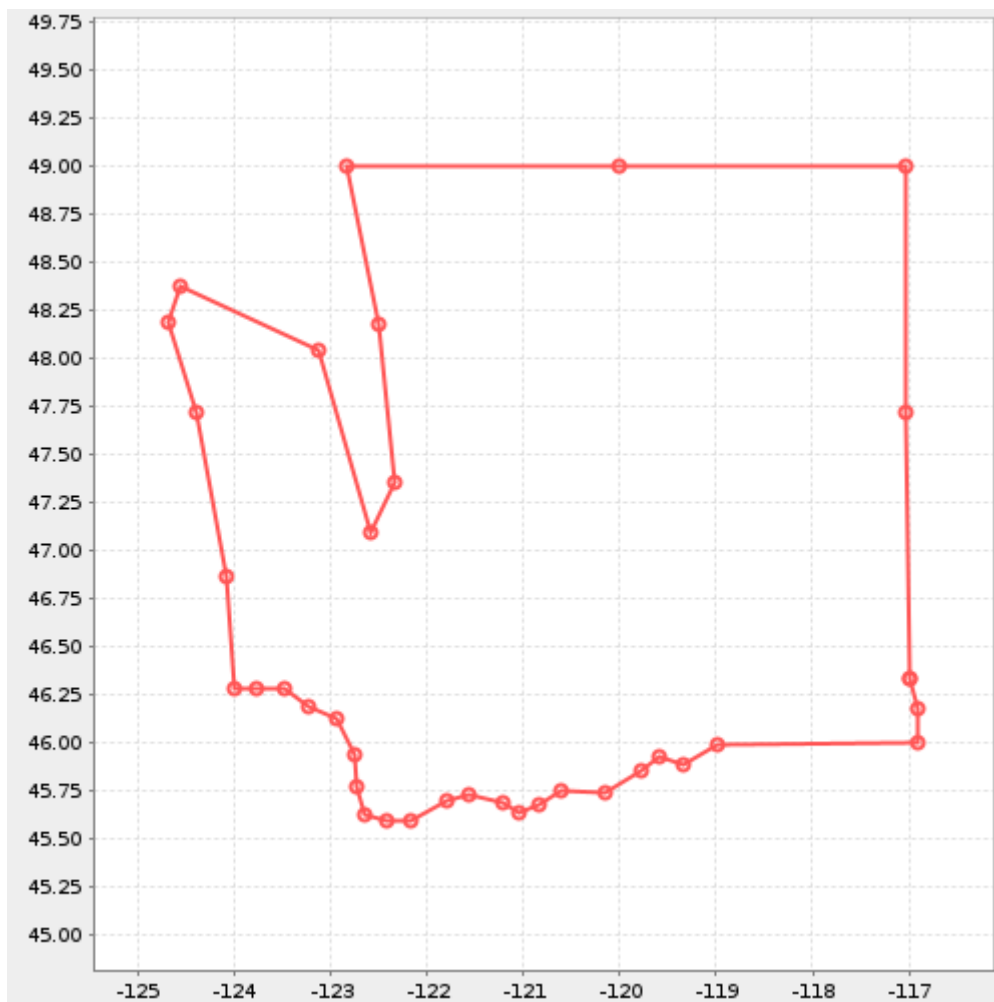
Plot a List of Geometries to an Image

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
BufferedImage image = Plot.plotToImage(geometries)
```

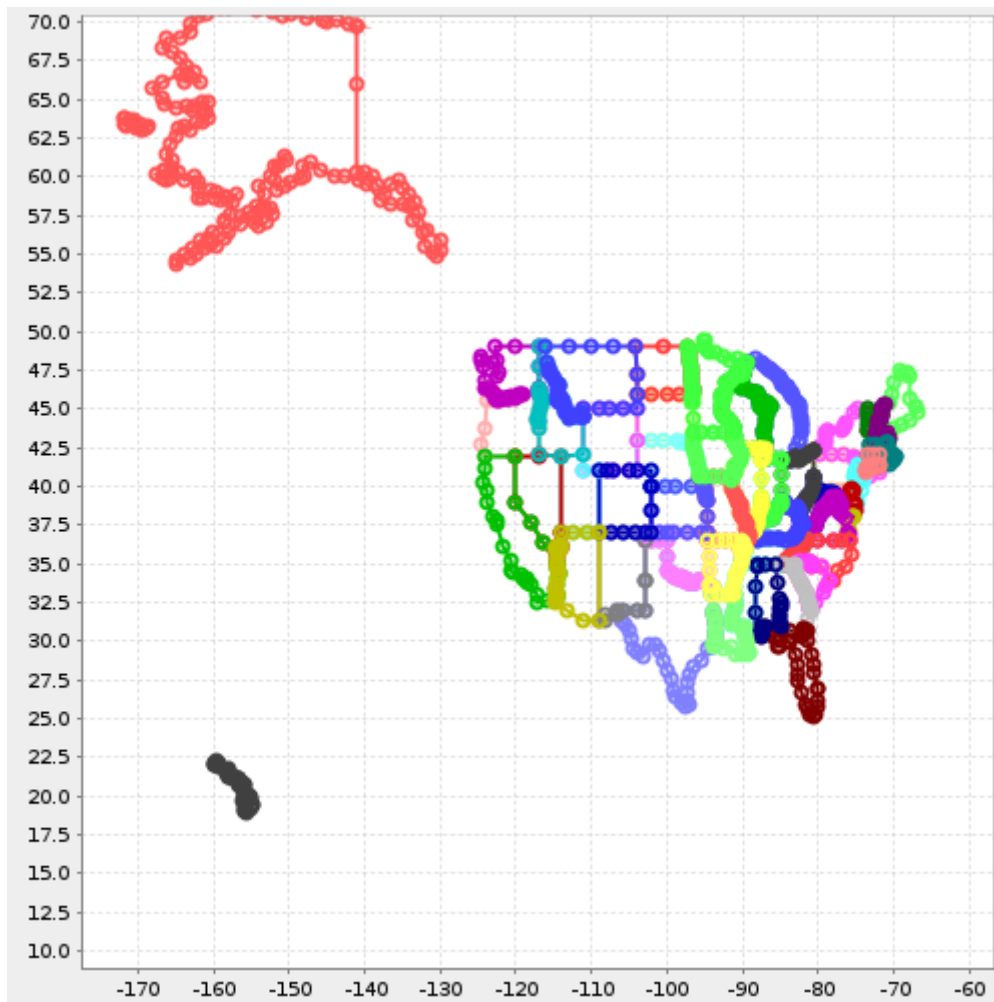
Plot a Feature to an Image

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



Plot a Layer to an Image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
BufferedImage image = Plot.plotToImage(layer, bounds: layer.bounds)
```



Reading Maps

The IO module can read Maps from JSON or XML documents.

Finding Map Readers

List all Map Readers

```
List<MapReader> readers = MapReaders.list()
readers.each { MapReader reader ->
    println reader.name
}
```

```
xml
json
```

Find a Map Reader

```
MapReader reader = MapReaders.find("json")
println reader.name
```

JSON

JSON Map Format

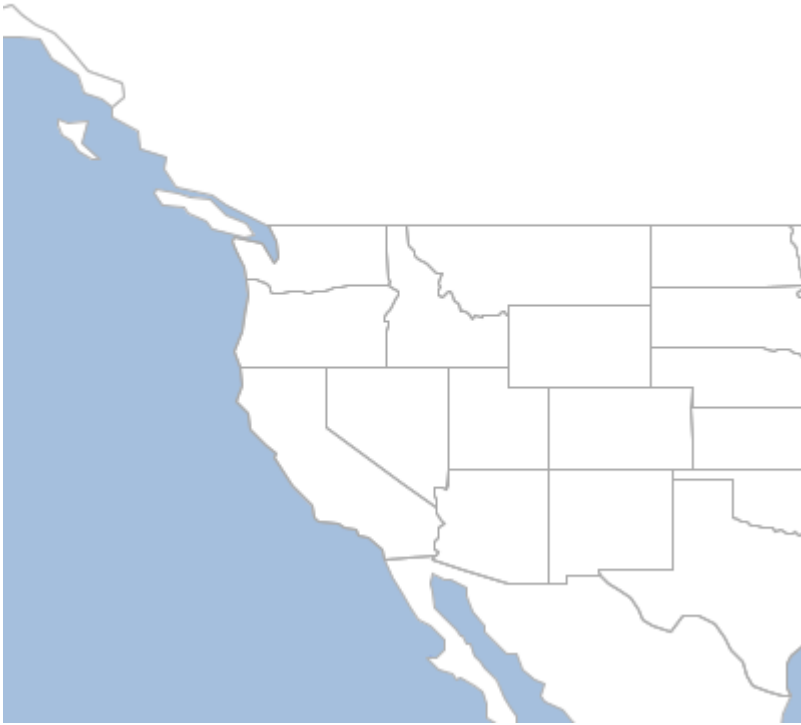
```
{
  "width": 400,
  "height": 400,
  "type": "png",
  "backgroundColor": "blue",
  "fixAspectRatio": true,
  "proj": "EPSG:4326",
  "bounds": {
    "minX": -135.911779,
    "minY": 36.993573,
    "maxX": -96.536779,
    "maxY": 51.405899
  },
  "layers": [
    {"layertype": "layer", "file": "shps/states.shp"}
  ]
}
```

- **width** = The map width is optional and defaults to 600.
- **height** = The map height is optional and defaults to 400.
- **type** = The image type (png, jpeg, gif) is optional and defaults to png.
- **backgroundColor** = The map background color is optional and transparent by default.
- **fixAspectRatio** = Whether to fix the aspect ratio or not. It is optional and the default is true.
- **proj** = The optional map projection. The default is determined by the layers or the bounds.
- **bounds** = The optional map bounds. The default is the full extent of the layers.
- **layers** = The only required property. A list of layer configurations.
 - **layertype** = The layer type is required. Values can be layer (vector), raster, or tile.
 - **layername** = The name of the layer is optional and defaults to first layer.
 - **style** = The layer style is optional but can be a SLD or CSS file.

Read a Map from a JSON String.

```
String json = ""{
    "width": 400,
    "height": 400,
    "type": "png",
    "backgroundColor": "blue",
    "proj": "EPSG:4326",
    "bounds": {
        "minX": -135.911779,
        "minY": 36.993573,
        "maxX": -96.536779,
        "maxY": 51.405899
    },
    "layers": [
        {
            "layertype": "layer",
            "dbtype": "geopkg",
            "database": "src/main/resources/data.gpkg",
            "layername": "ocean",
            "style": "src/main/resources/ocean.sld"
        },
        {
            "layertype": "layer",
            "dbtype": "geopkg",
            "database": "src/main/resources/data.gpkg",
            "layername": "countries",
            "style": "src/main/resources/countries.sld"
        },
        {
            "layertype": "layer",
            "dbtype": "geopkg",
            "database": "src/main/resources/data.gpkg",
            "layername": "states",
            "style": "src/main/resources/states.sld"
        }
    ]
}
""

MapReader mapReader = new JsonMapReader()
Map map = mapReader.read(json)
BufferedImage image = map.renderToImage()
```



XML

```
String xml = ""<map>
<width>400</width>
<height>400</height>
<type>png</type>
<proj>EPSG:4326</proj>
<backgroundColor>blue</backgroundColor>
<fixAspectRatio>true</fixAspectRatio>
<layers>
  <layer>
    <layertype>layer</layertype>
    <dbtype>geopkg</dbtype>
    <database>src/main/resources/data.gpkg</database>
    <layername>ocean</layername>
    <style>src/main/resources/ocean.sld</style>
  </layer>
  <layer>
    <layertype>layer</layertype>
    <dbtype>geopkg</dbtype>
    <database>src/main/resources/data.gpkg</database>
    <layername>countries</layername>
    <style>src/main/resources/countries.sld</style>
  </layer>
  <layer>
    <layertype>layer</layertype>
    <dbtype>geopkg</dbtype>
    <database>src/main/resources/data.gpkg</database>
    <layername>states</layername>
    <style>src/main/resources/states.sld</style>
  </layer>
</layers>
<bounds>
  <minX>-135.911779</minX>
  <minY>36.993573</minY>
  <maxX>-96.536779</maxX>
  <maxY>51.405899</maxY>
</bounds>
</map>
""

MapReader mapReader = new XmlMapReader()
Map map = mapReader.read(xml)
BufferedImage image = map.renderToImage()
```

