Table of Contents

Style Recipes

Styles are found in the **geoscript.style** package.

Creating Strokes

Create a Stroke Symbolizer with a Color

```
Stroke stroke = new Stroke("#1E90FF")
```

Create a Stroke Symbolizer with a Color and Width

```
Stroke stroke = new Stroke("#1E90FF", 0.5)
```

Create a Stroke Symbolizer with casing

```
Symbolizer stroke = new Stroke(color: "#333333", width: 5, cap: "round").zindex(0) +
        new Stroke(color: "#6699FF", width: 3, cap: "round").zindex(1)
```

Create a Stroke Symbolizer with Dashes

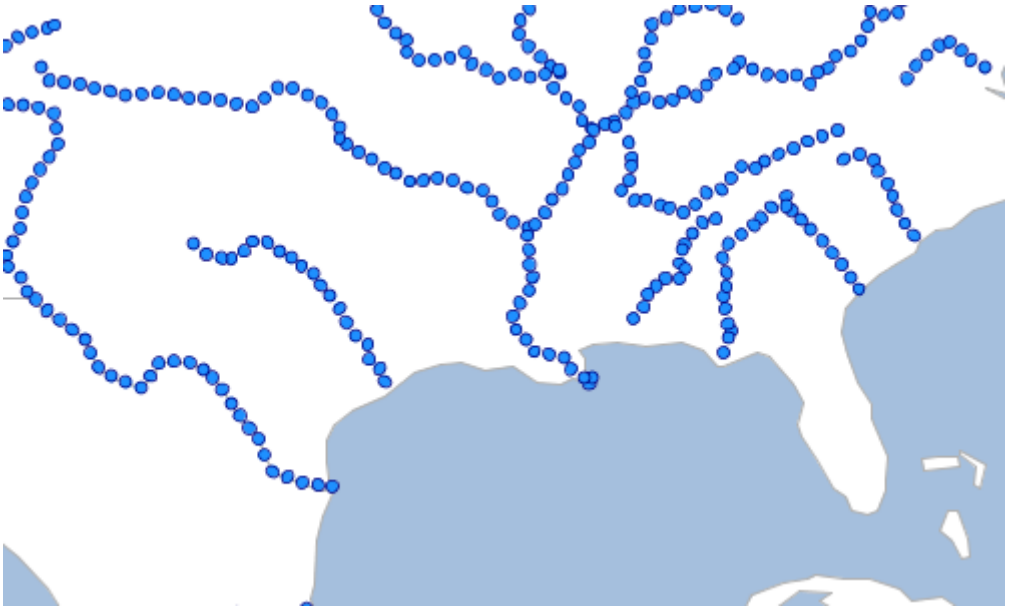
```
Stroke stroke = new Stroke("#1E90FF", 0.75, [5,5], "round", "bevel")
```

Create a Stroke Symbolizer with railroad Hatching

```
Symbolizer stroke = new Stroke("#1E90FF", 1) + new Hatch("vertline", new Stroke
("#1E90FF", 0.5), 6).zindex(1)
```

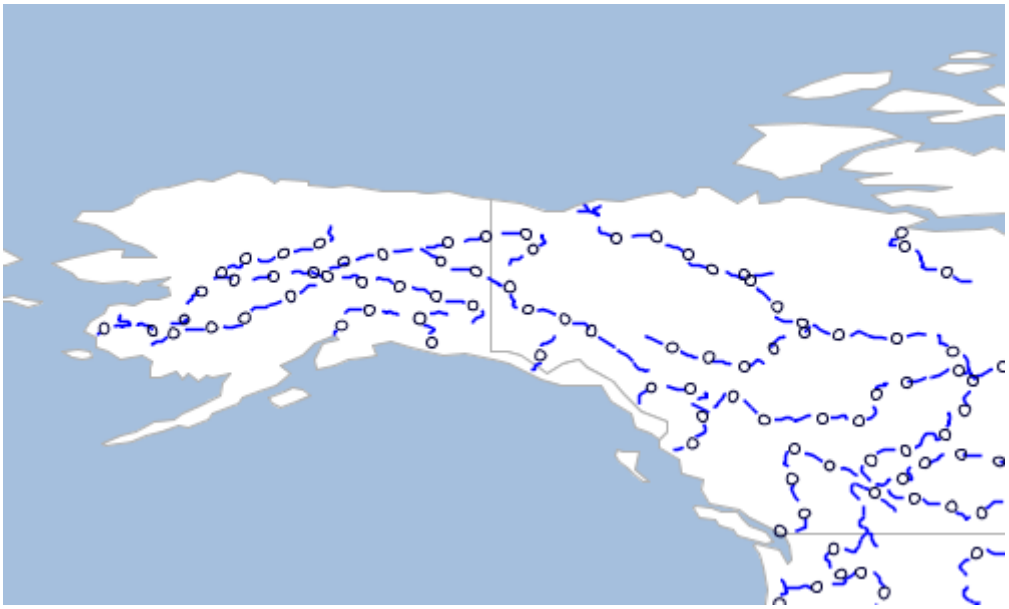
Create a Stroke Symbolizer with spaced Shape symbols

```
Symbolizer stroke = new Stroke(width: 0, dash: [4, 4]).shape(new Shape("#1E90FF", 6,
"circle").stroke("navy", 0.75))
```



Create a Stroke Symbolizer with alternating spaced Shape symbols

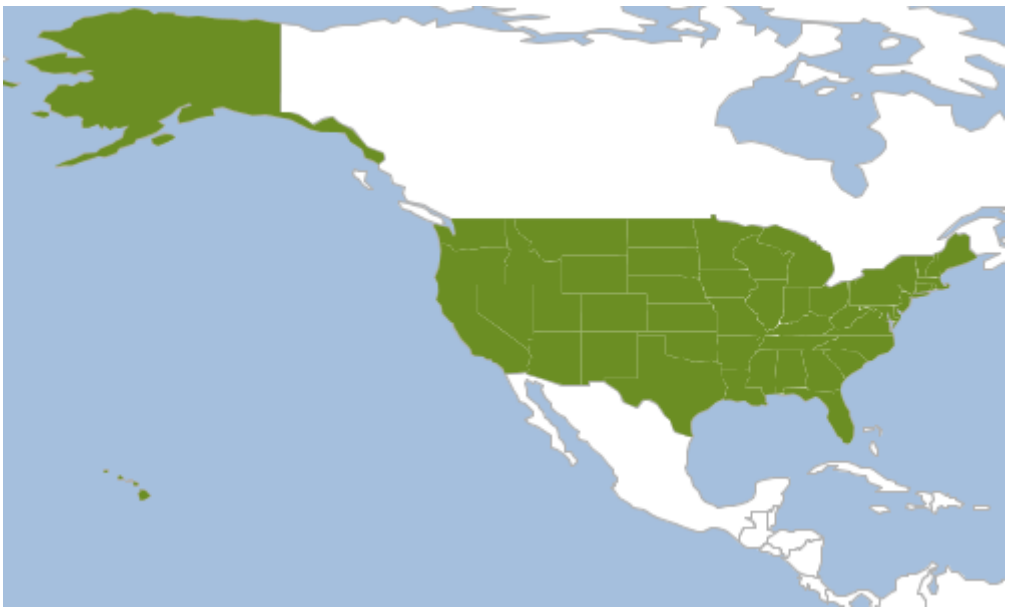
```
Symbolizer stroke = new Stroke("#0000FF", 1, [10,10]).zindex(0) + new Stroke(null, 0,
[[5,15],7.5])
        .shape(new Shape(null, 5, "circle").stroke("#000033",1)).zindex(1)
```



Creating Fills

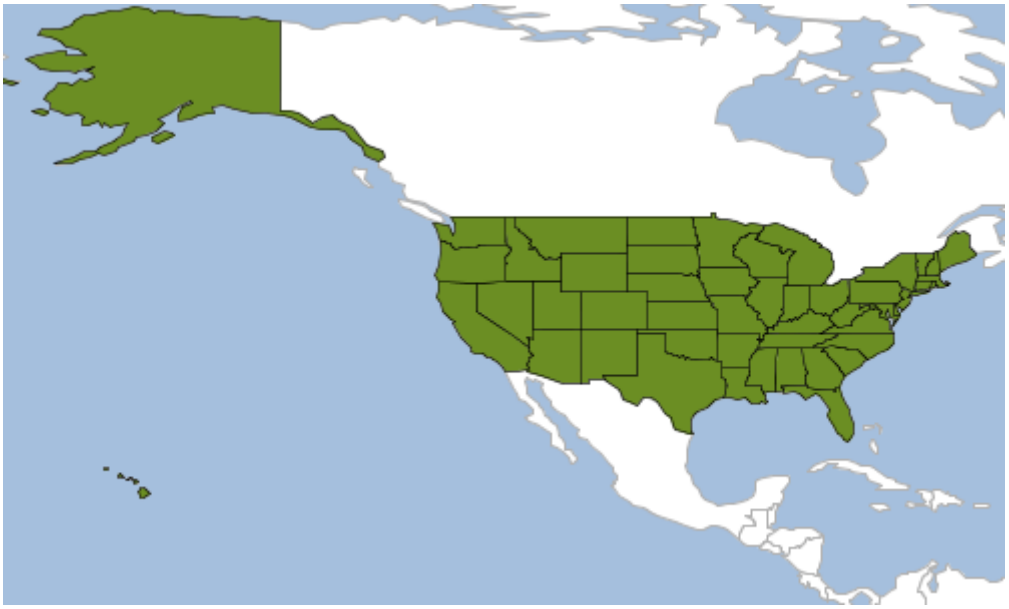
Create a Fill Symbolizer with a Color

```
Fill fill = new Fill("#6B8E23")
```



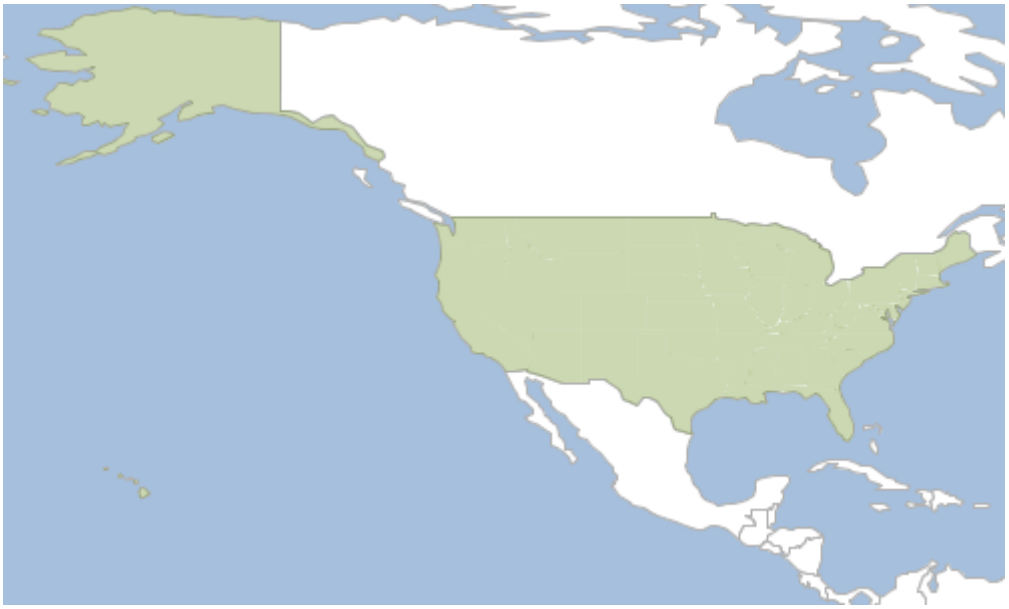
Create a Fill Symbolizer with a Color and a Stroke

```
Symbolizer symbolizer = new Fill("#6B8E23") + new Stroke("black", 0.1)
```



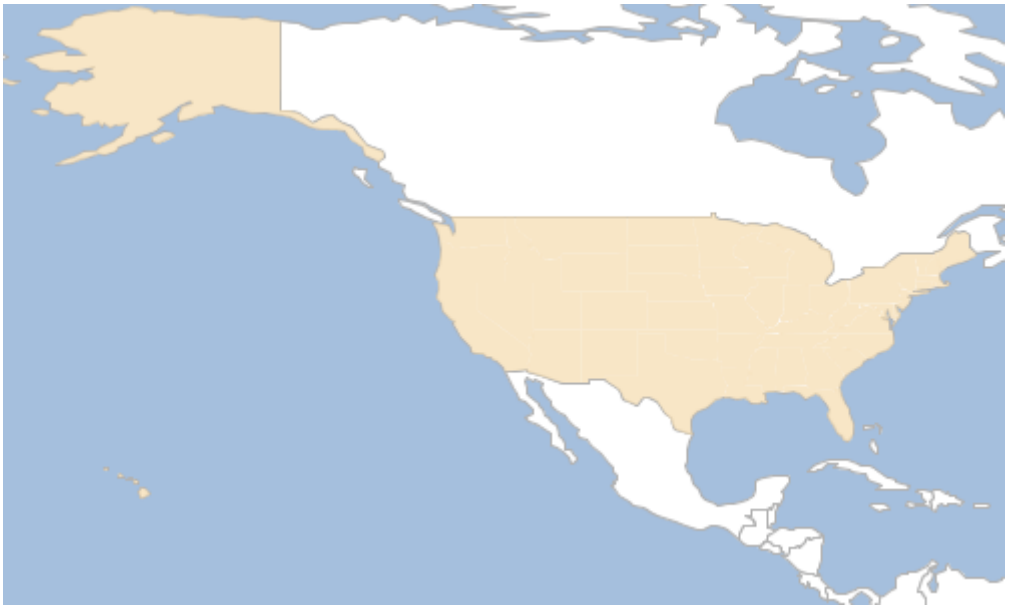
Create a Fill Symbolizer with a Color and Opacity

```
Fill fill = new Fill("#6B8E23", 0.35)
```



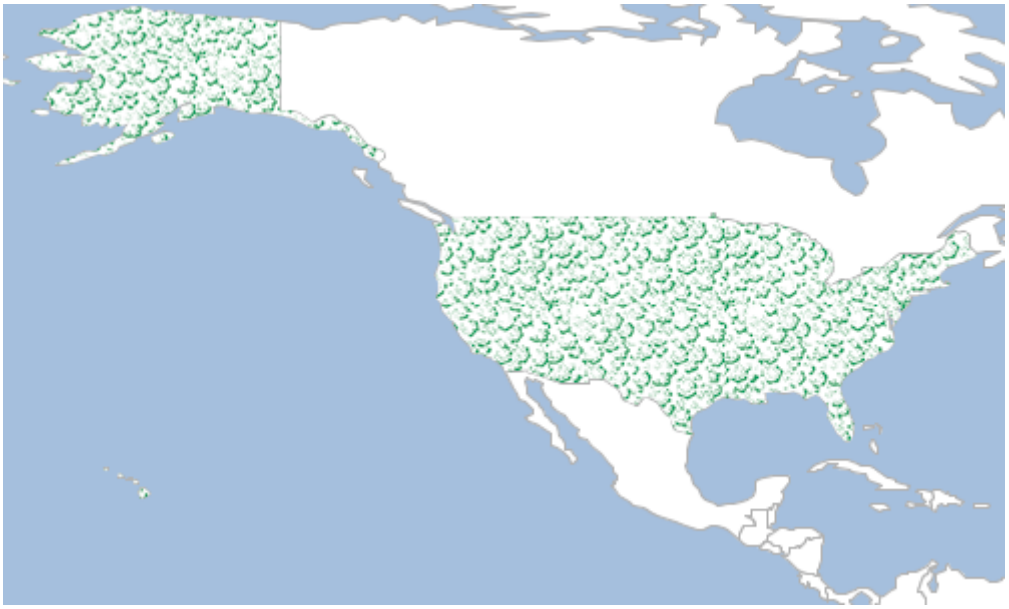
Create a Fill Symbolizer from named parameters

```
Fill fill = new Fill(color: "wheat", opacity: 0.75)
```



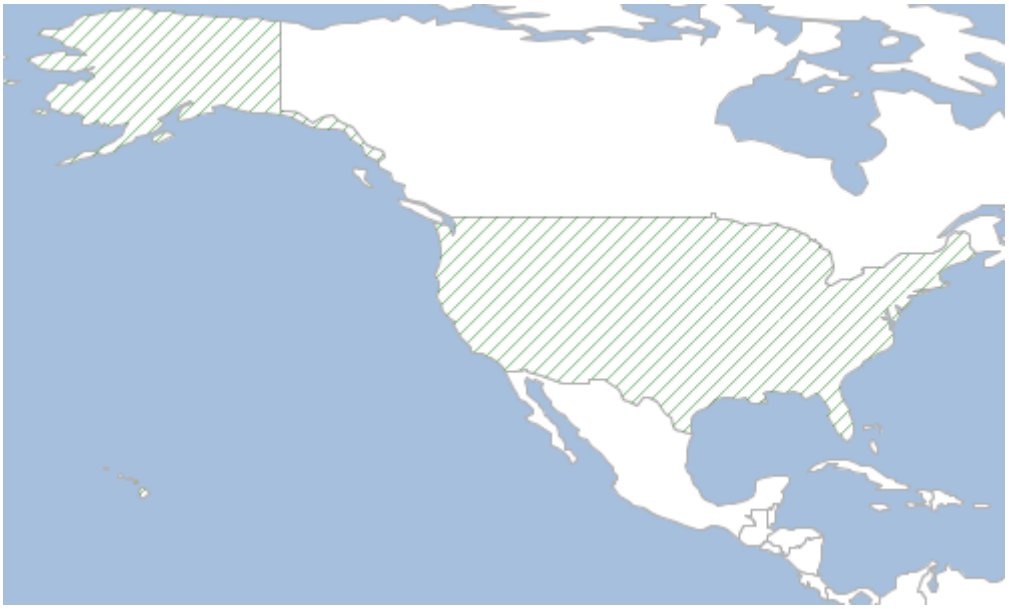
Create a Fill Symbolizer with an Icon

```
Fill fill = new Fill("green").icon('src/main/resources/trees.png', 'image/png')
```



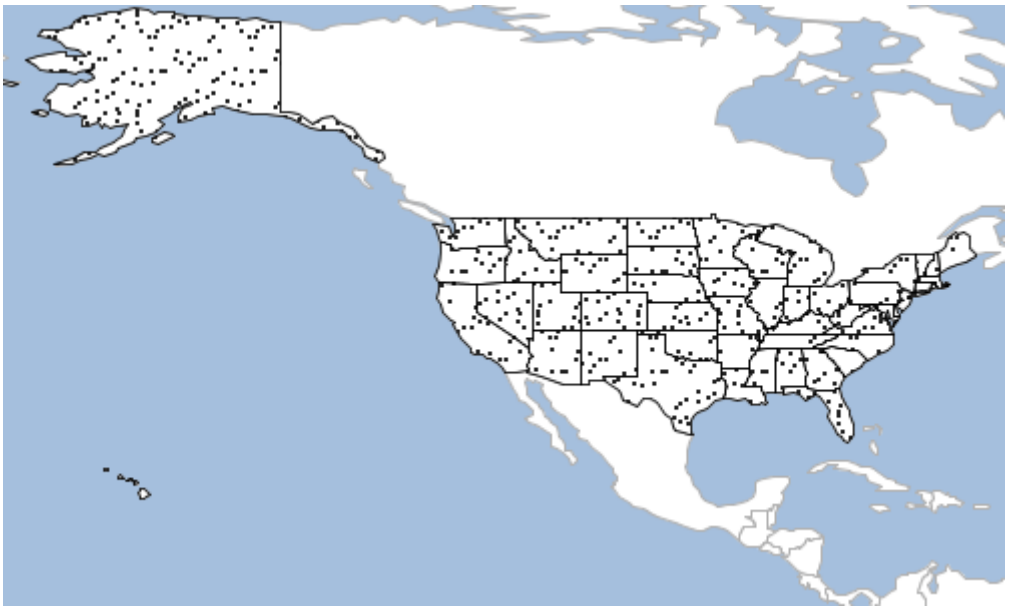
Create a Fill Symbolizer with a Hatch

```
Fill fill = new Fill("green").hatch("slash", new Stroke("green", 0.25), 8)
```



Create a Fill Symbolizer with a random fill

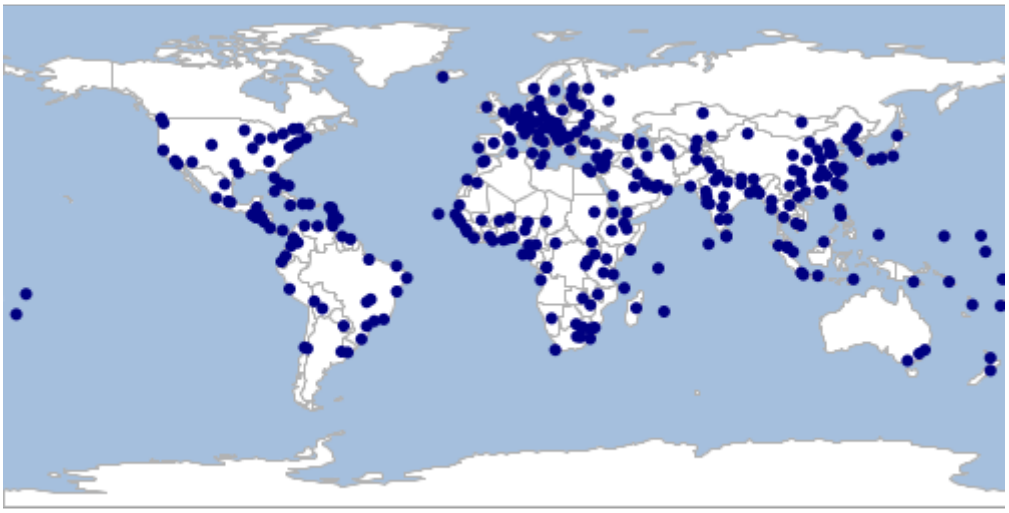
```
Symbolizer symbolizer = new Fill("white").hatch("circle", new Fill("black"), 2).
random(
    random: "free",
    seed: 0,
    symbolCount: 50,
    tileSize: 50,
    rotation: "none"
) + new Stroke("black", 0.25)
```



Creating Shapes

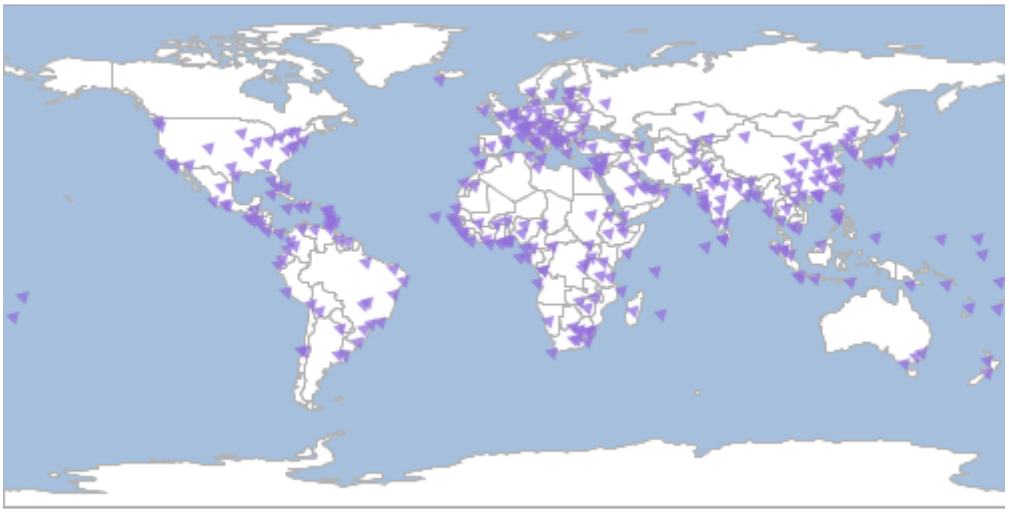
Create a Shape Symbolizer with a Color

```
Shape shape = new Shape("navy")
```



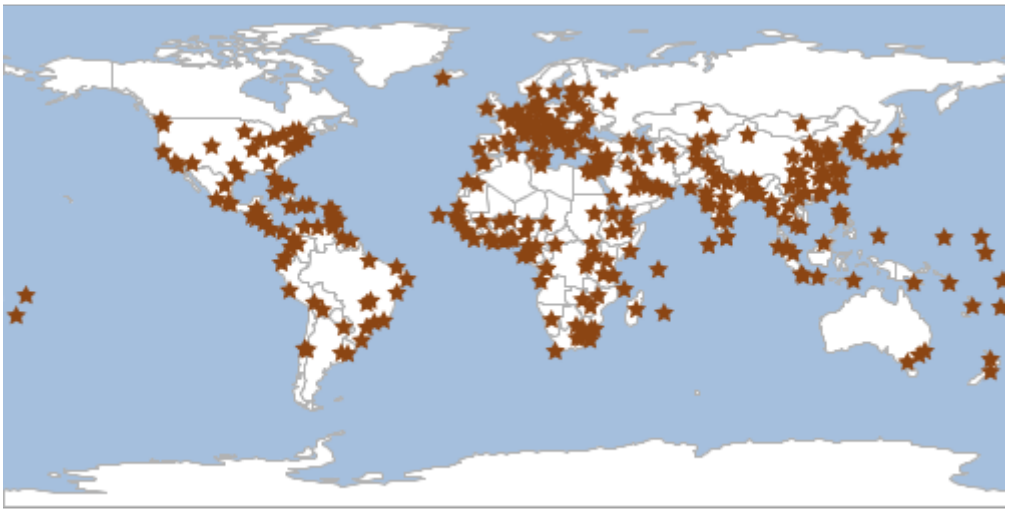
Create a Shape Symbolizer with a color, size, type, opacity and angle

```
Shape shape = new Shape("#9370DB", 8, "triangle", 0.75, 45)
```



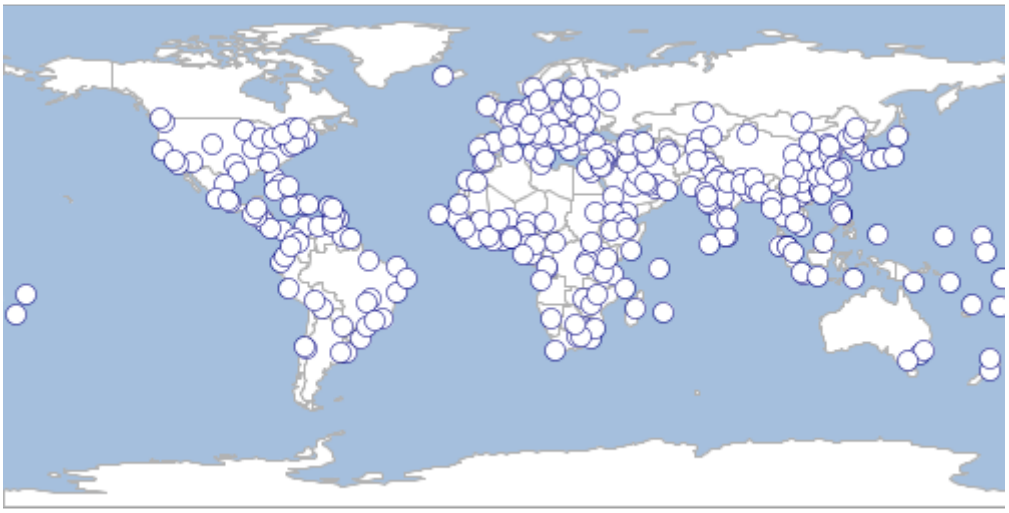
Create a Shape Symbolizer with named parameters

```
Shape shape = new Shape(color: "#8B4513", size: 10, type: "star", opacity: 1.0,
rotation: 0)
```



Create a Shape Symbolizer with Stroke outline

```
Symbolizer symbolizer = new Shape("white", 10).stroke("navy", 0.5)
```



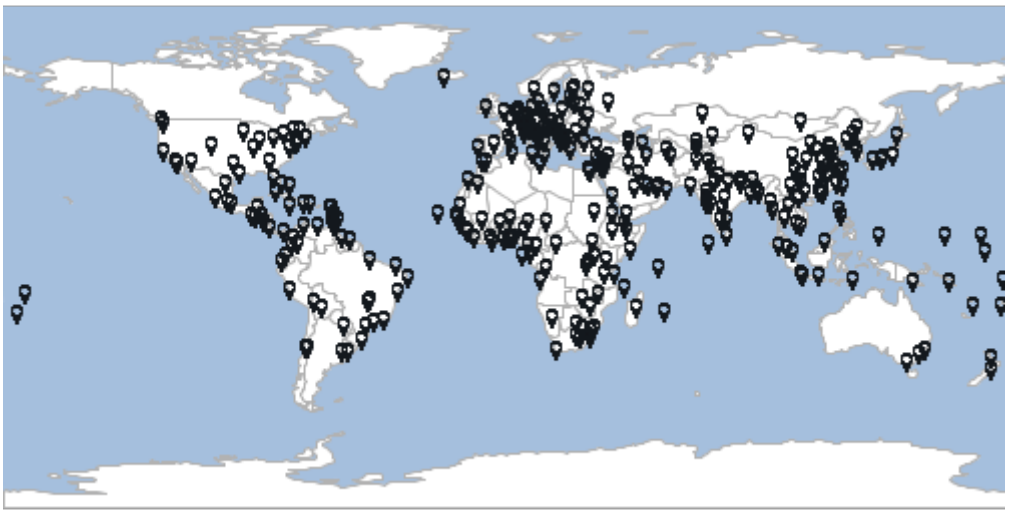
Creating Icons

Create an Icon Symbolizer

```
Symbolizer symbolizer = new Icon("src/main/resources/place.png", "image/png", 12)
```

Create an Icon Symbolizer

```
Symbolizer symbolizer = new Icon(url: "src/main/resources/place.png", format:
"image/png", size: 10)
```



Creating Labels

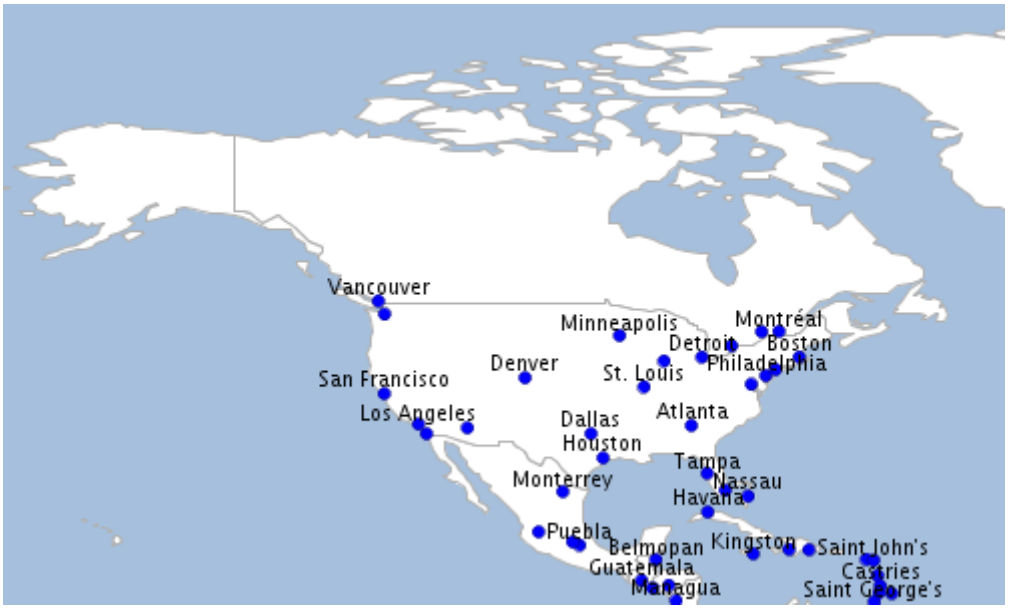
Create a Label for a Point Layer

```
Symbolizer symbolizer = new Shape("blue", 6).stroke("navy", 0.5) + new Label("NAME"
).point(
        [0.5,0.5],  ①
        [0, 5.0],   ②
        0           ③
)
```

① anchor

② displacement

③ rotation



Create a Label for a Polygon Layer

```
Symbolizer symbolizer = new Fill("white") + new Stroke("black", 0.1) + new Label
("NAME_1")
    .point(anchor: [0.5,0.5])
    .polygonAlign("mbr")
```

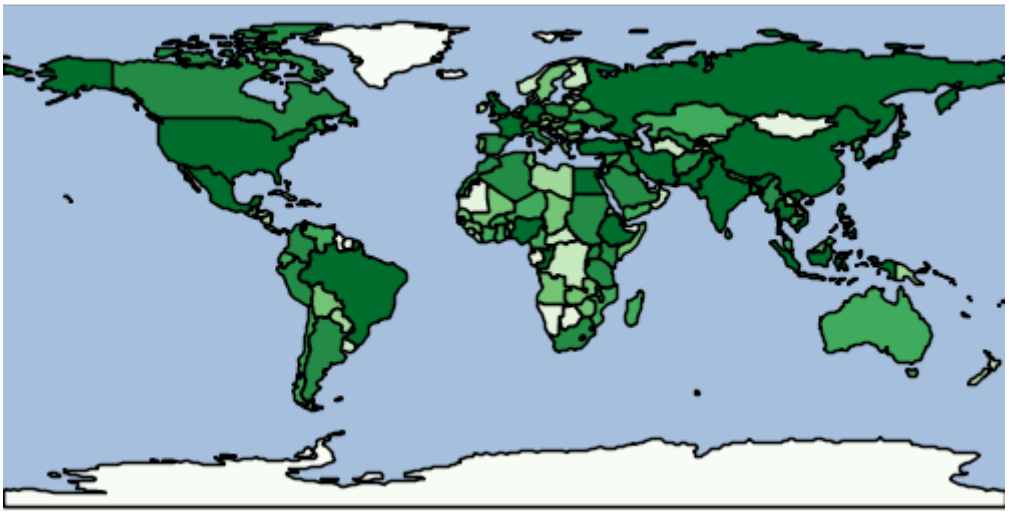
Create a Label for a Line Layer

```
Symbolizer symbolizer = new Stroke("blue", 0.75)  + new Label("name")
        .fill(new Fill("navy"))
        .linear(follow: true, offset: 50, displacement: 200, repeat: 150
).maxDisplacement(400).maxAngleDelta(90)
        .halo(new Fill("white"), 2.5)
        .font(new Font(size: 10, weight: "bold"))
```

Creating Gradients

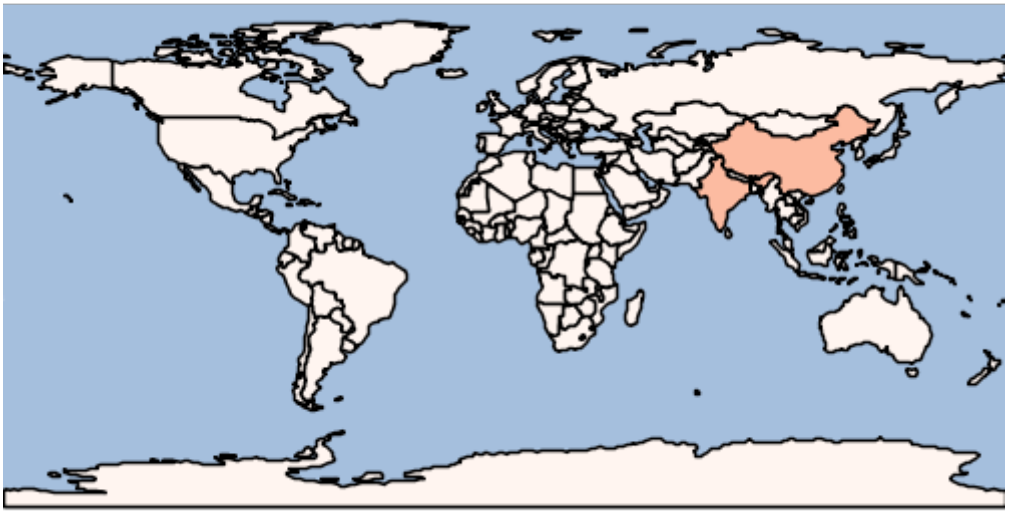
Create a Gradient Symbolizer from a Layer's Field using quantile method

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
Gradient gradient = new Gradient(countries, "PEOPLE", "quantile", 8, "Greens")
countries.style = gradient
```



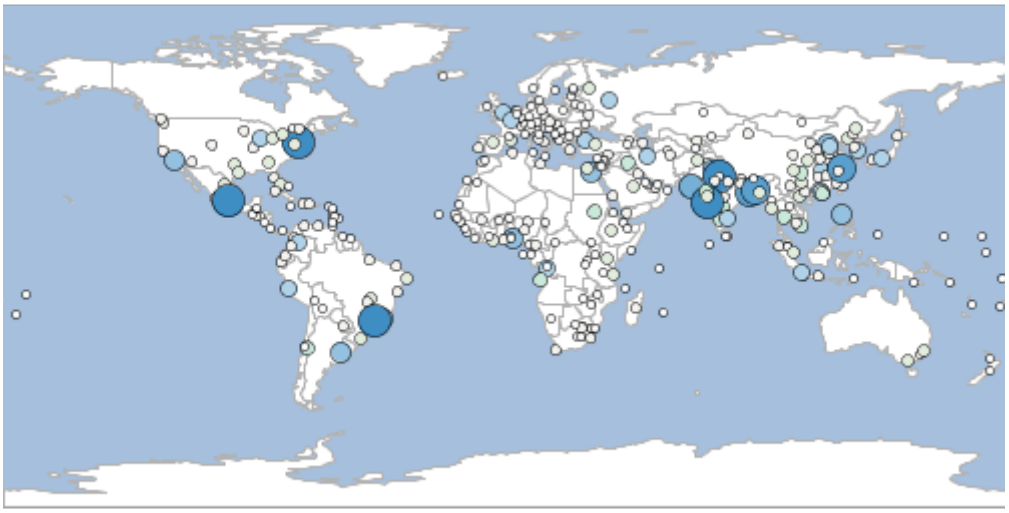
Create a Gradient Symbolizer from a Layer's Field using equal interval method

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
Gradient gradient = new Gradient(countries, "PEOPLE", "equalinterval", 3, "Reds")
countries.style = gradient
```



Create a custom Gradient Symbolizer between Symbolizers and values

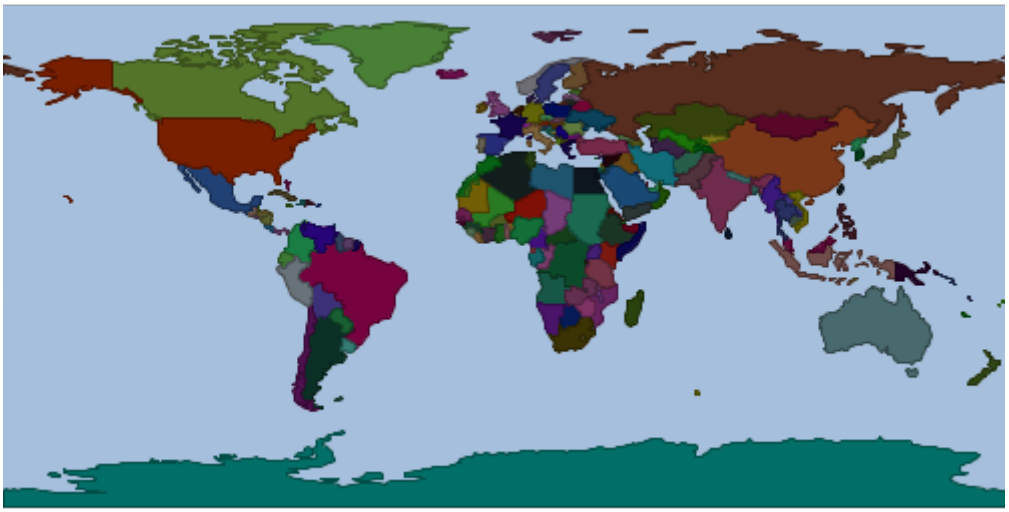
```
Gradient gradient = new Gradient(
        new Property("POP2020"),
        [0, 10000, 20000, 30000],
        [
                new Shape("white", 4).stroke("black", 0.5),
                new Shape("#b0d2e8", 8).stroke("black", 0.5),
                new Shape("#3e8ec4", 16).stroke("black", 0.5),
                new Shape("#08306b", 24).stroke("black", 0.5)
        ],
        5,
        "linear"
)
```



Creating Unique Values

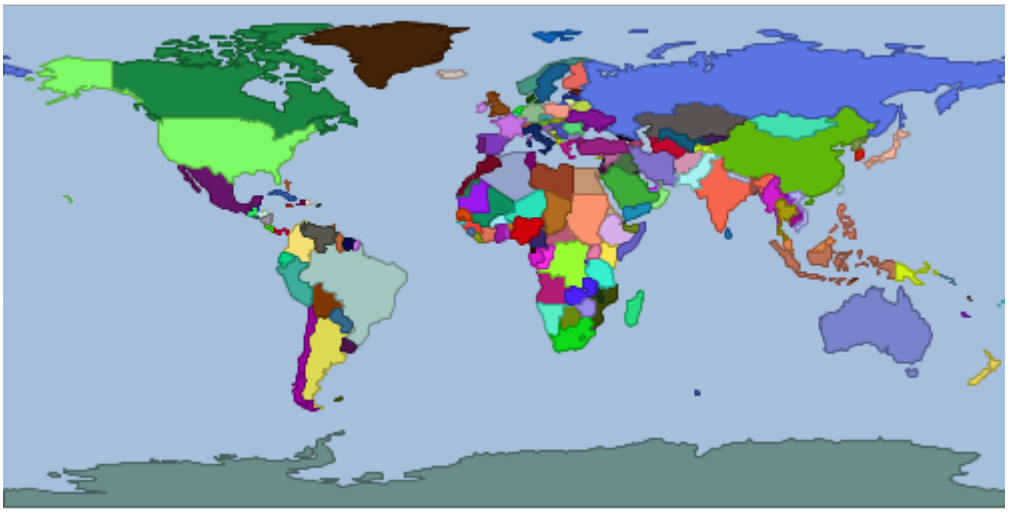
Create a Unique Values Symbolizer from a Layer's Field

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
UniqueValues uniqueValues = new UniqueValues(countries, "NAME")
countries.style = uniqueValues
```



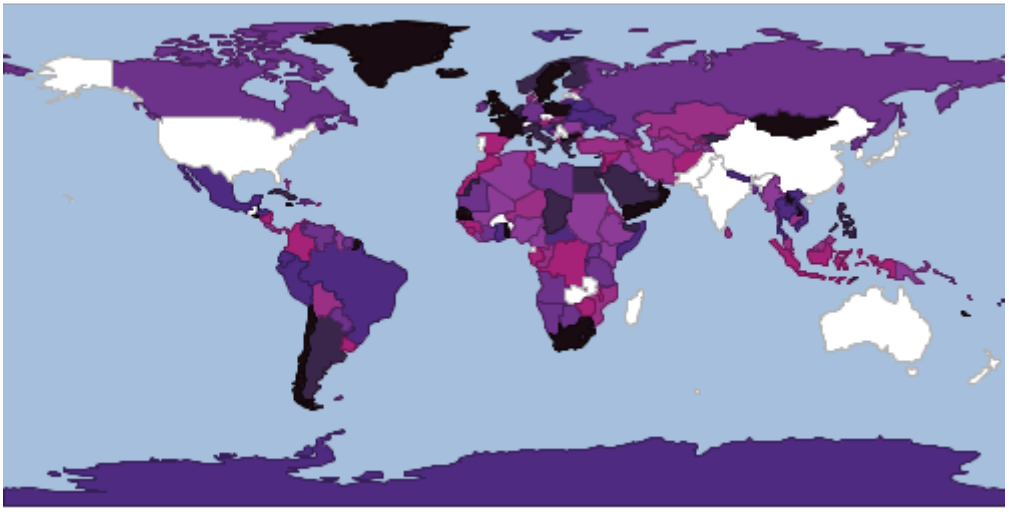
Create a Unique Values Symbolizer from a Layer's Field and a Closure

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
UniqueValues uniqueValues = new UniqueValues(countries, "NAME", {int index, String
value -> Color.getRandom()})
countries.style = uniqueValues
```



Create a Unique Values Symbolizer from a Layer's Field and a color palette

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
UniqueValues uniqueValues = new UniqueValues(countries, "NAME",
"LightPurpleToDarkPurpleHeatMap")
countries.style = uniqueValues
```



Reading and Writing Styles

Style Readers and Writers are found in the **geoscript.style.io** package.

Finding Style Readers and Writers

List all Style Writers

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

```
SLDWriter
ColorTableWriter
YSLDWriter
```

Find a Style Writer

```
Writer writer = Writers.find("sld")
println writer.class.simpleName
```

```
SLDWriter
```

List all Style Readers

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

```
SLDReader
CSSReader
ColorTableReader
YSLDReader
SimpleStyleReader
```

Find a Style Reader

```
Reader reader = Readers.find("sld")
println reader.class.simpleName
```

```
SLDReader
```

SLD

GeoScript Groovy can read and write Style Layer Descriptor (SLD) documents.

Write a Symbolizer to SLD

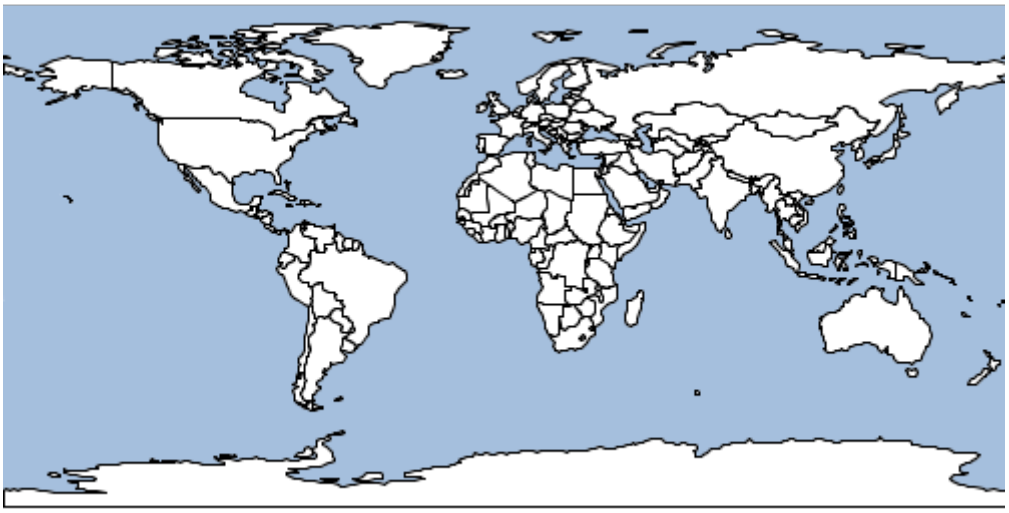
```
Symbolizer symbolizer = new Fill("white") + new Stroke("black", 0.5)
SLDWriter writer = new SLDWriter()
String sld = writer.write(symbolizer)
println sld
```

```
<?xml version="1.0" encoding="UTF-8"?>
<sld:StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
xmlns:sld="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml" version="1.0.0">
  <sld:UserLayer>
    <sld:LayerFeatureConstraints>
      <sld:FeatureTypeConstraint/>
    </sld:LayerFeatureConstraints>
    <sld:UserStyle>
      <sld:Name>Default Styler</sld:Name>
      <sld:FeatureTypeStyle>
        <sld:Name>name</sld:Name>
        <sld:Rule>
          <sld:PolygonSymbolizer>
            <sld:Fill>
              <sld:CssParameter name="fill">#ffffff</sld:CssParameter>
            </sld:Fill>
          </sld:PolygonSymbolizer>
          <sld:LineSymbolizer>
            <sld:Stroke>
              <sld:CssParameter name="stroke-width">0.5</sld:CssParameter>
            </sld:Stroke>
          </sld:LineSymbolizer>
        </sld:Rule>
      </sld:FeatureTypeStyle>
    </sld:UserStyle>
  </sld:UserLayer>
</sld:StyledLayerDescriptor>
```

Read a Style from an SLD String

```
        String sld = """<?xml version="1.0" encoding="UTF-8"?>
<sld:StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
xmlns:sld="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml" version="1.0.0">
  <sld:UserLayer>
    <sld:LayerFeatureConstraints>
      <sld:FeatureTypeConstraint/>
    </sld:LayerFeatureConstraints>
    <sld:UserStyle>
      <sld:Name>Default Styler</sld:Name>
      <sld:FeatureTypeStyle>
        <sld:Name>name</sld:Name>
        <sld:Rule>
          <sld:PolygonSymbolizer>
            <sld:Fill>
              <sld:CssParameter name="fill">#ffffff</sld:CssParameter>
            </sld:Fill>
          </sld:PolygonSymbolizer>
          <sld:LineSymbolizer>
            <sld:Stroke>
              <sld:CssParameter name="stroke">#000000</sld:CssParameter>
              <sld:CssParameter name="stroke-width">0.5</sld:CssParameter>
            </sld:Stroke>
          </sld:LineSymbolizer>
        </sld:Rule>
      </sld:FeatureTypeStyle>
    </sld:UserStyle>
  </sld:UserLayer>
</sld:StyledLayerDescriptor>
"""
        SLDReader reader = new SLDReader()
        Style style = reader.read(sld)

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



CSS

GeoScript Groovy can only read CSS documents.

Read a Style from an CSS String

```
        String css = """
* {
   fill: #eeeeee;
   fill-opacity: 1.0;
   stroke: #000000;
   stroke-width: 1.2;
}
"""
        CSSReader reader = new CSSReader()
        Style style = reader.read(css)

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