Table of Contents

Carto Recipes

The Carto classes are in the **geoscript.carto** package.

The Carto package contains classes for creating cartographic documents. All items are added to the document with x and y coordinates whose origin is the upper left and width and a height.

Items

Adding a Map

Add a map

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
    layers: [ocean, countries],
    bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
    projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 40).map
(map))
            .build(outputStream)

}
```

Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
map	geoscript.render.Map	The Map to display

Adding an Overview Map

Add an overview map

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-166.436348,6.574916,-12.451973,60.715022, "EPSG:4326"
).reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)
Map overViewMap = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
        .create(pageSize)
        .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height - 1)
            .fillColor(Color.WHITE)
        )
        .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 40).map(map))
        .rectangle(new RectangleItem(20, 20, pageSize.width - 40, pageSize.height -
40))
        .overViewMap(new OverviewMapItem(40, pageSize.height - 240, 200, 200)
            .linkedMap(map)
            .overviewMap(overViewMap)
            .zoomIntoBounds(false)
        )
        .rectangle(new RectangleItem(40, pageSize.height - 240, 200,200))
        .build(outputStream)

}
```

Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
overviewMap	geoscript.render.Map	The overview Map
linkedMap	geoscript.render.Map	The Map the overview Map is linked to
areaStyle	geoscript.style.Style	The GeoScript style to display the rectangle
zoomIntoBounds	boolean	Whether to zoom into the bounds of the linked Map or not
scaleFactor	double	The scale factor for expanding the linked Map Bounds

Adding a Text

Add text

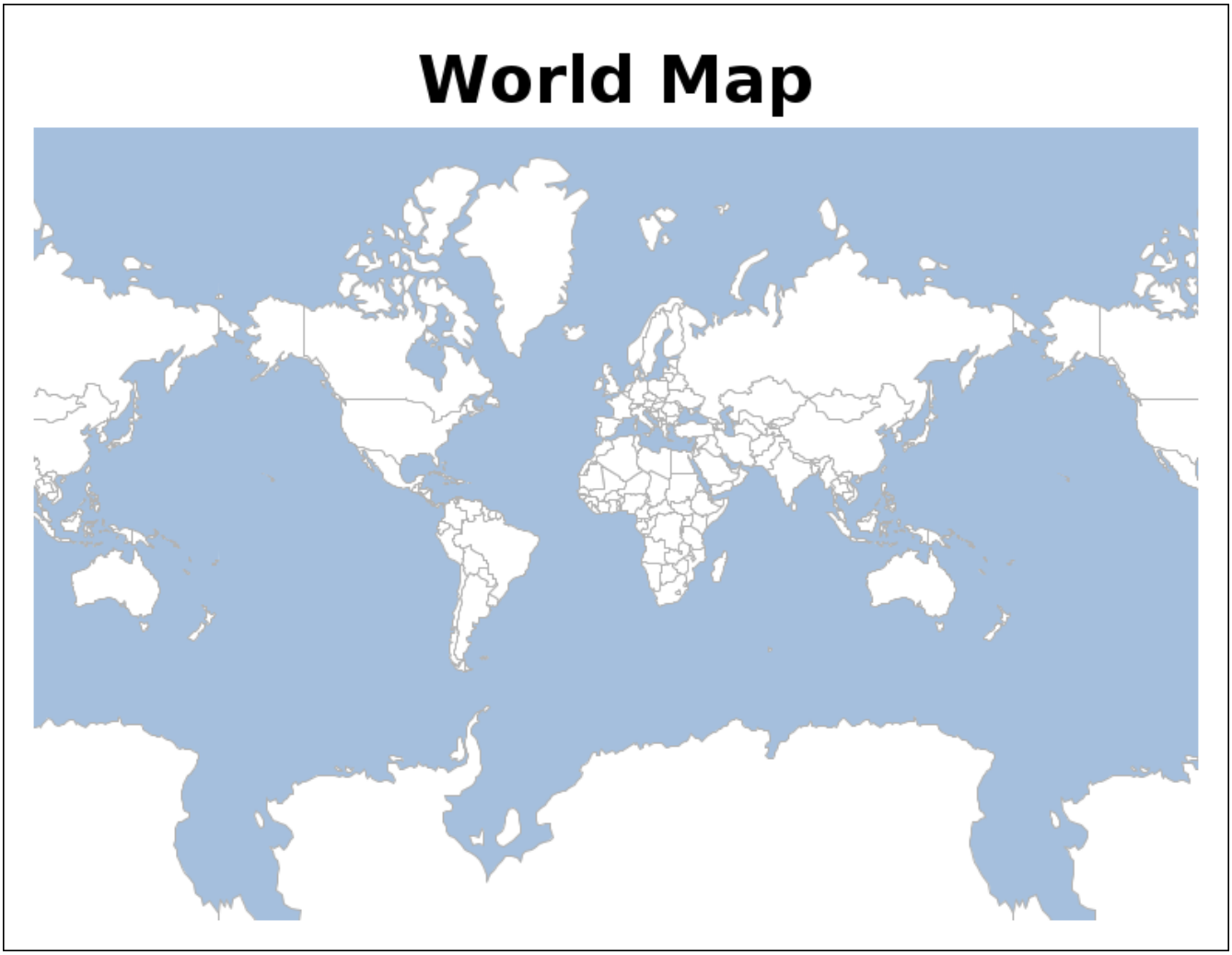
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .text(new TextItem(20,20, pageSize.width - 40, 60)
                .text("World Map")
                .font(new Font("Arial", Font.BOLD, 42))
                .verticalAlign(VerticalAlign.MIDDLE)
                .horizontalAlign(HorizontalAlign.CENTER)
            )
            .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map
(map))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
text	String	The text to display
color	java.awt.Color	The text Color
font	java.awt.Font	The text font
horizontalAlign	HorizontalAlign (LEFT, CENTER, RIGHT)	The horizontal alignment
verticalAlign	VerticalAlign (TOP, MIDDLE, BOTTOM)	The vertical alignment

Adding a Rectangle

Add a rectangle

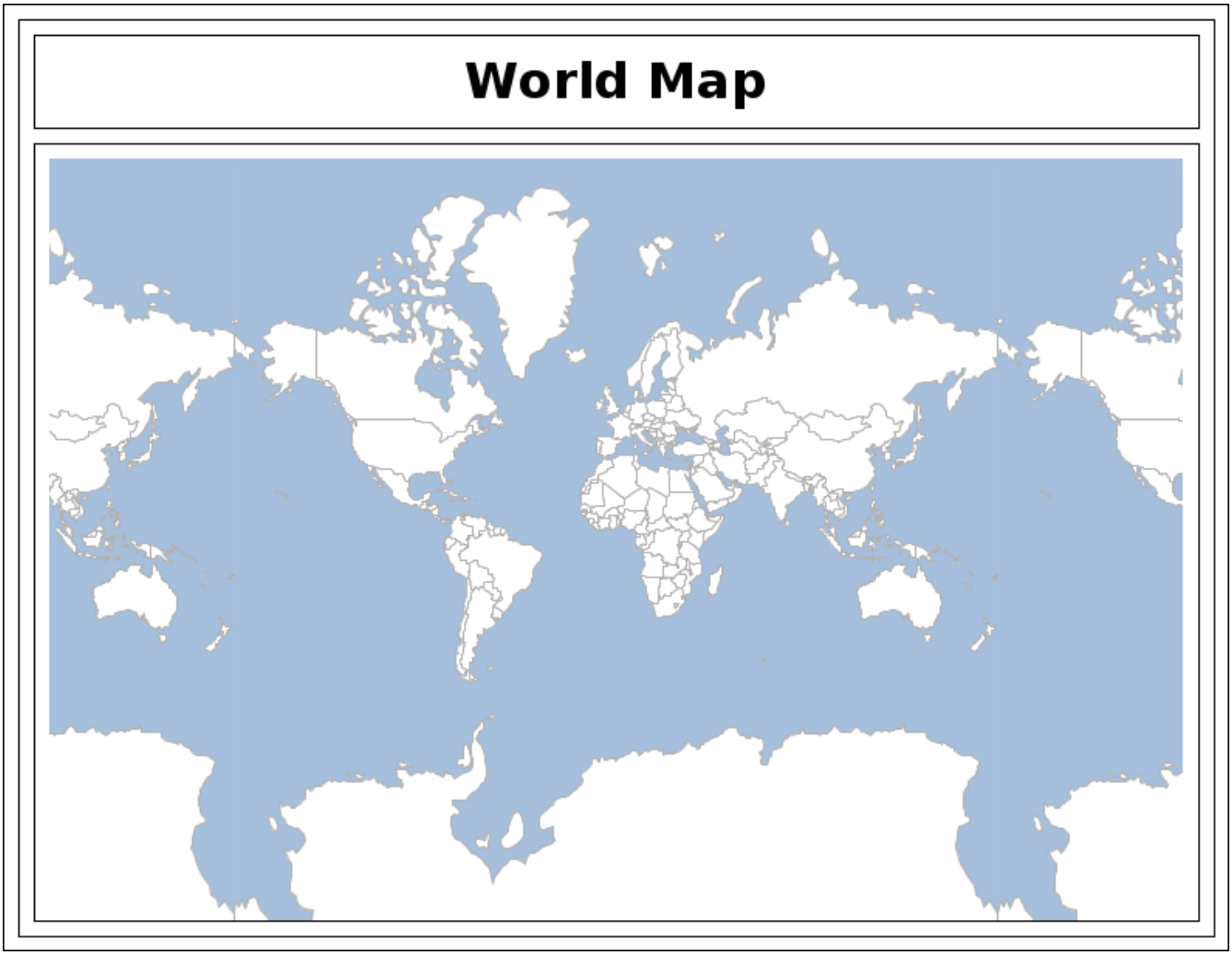
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .rectangle(new RectangleItem(10,10, pageSize.width - 20, pageSize.height -
20))
            .rectangle(new RectangleItem(20,20, pageSize.width - 40, 60))
            .rectangle(new RectangleItem(20,90, pageSize.width - 40, pageSize.height -
110))
            .text(new TextItem(20,20, pageSize.width - 40, 60)
                .text("World Map")
                .font(new Font("Arial", Font.BOLD, 32))
                .verticalAlign(VerticalAlign.MIDDLE)
                .horizontalAlign(HorizontalAlign.CENTER)
            )
            .map(new MapItem(30, 100, pageSize.width - 60, pageSize.height - 120).map
(map))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
strokeColor	java.awt.Color	The outline Color
fillColor	java.awt.Color	The fill Color
strokeWidth	float	The width of the stroke

Adding a North Arrow

Add a north arrow

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
        .create(pageSize)
        .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height - 1)
            .fillColor(Color.WHITE)
        )
        .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 40).map(map))
        .northArrow(new NorthArrowItem(pageSize.width - 60, pageSize.height - 100, 40,
80)
            .font(new Font("Arial", Font.BOLD, 24))
            .drawText(true))
        .build(outputStream)

}
```

Adding a NESW North Arrow

Add a NESW north arrow

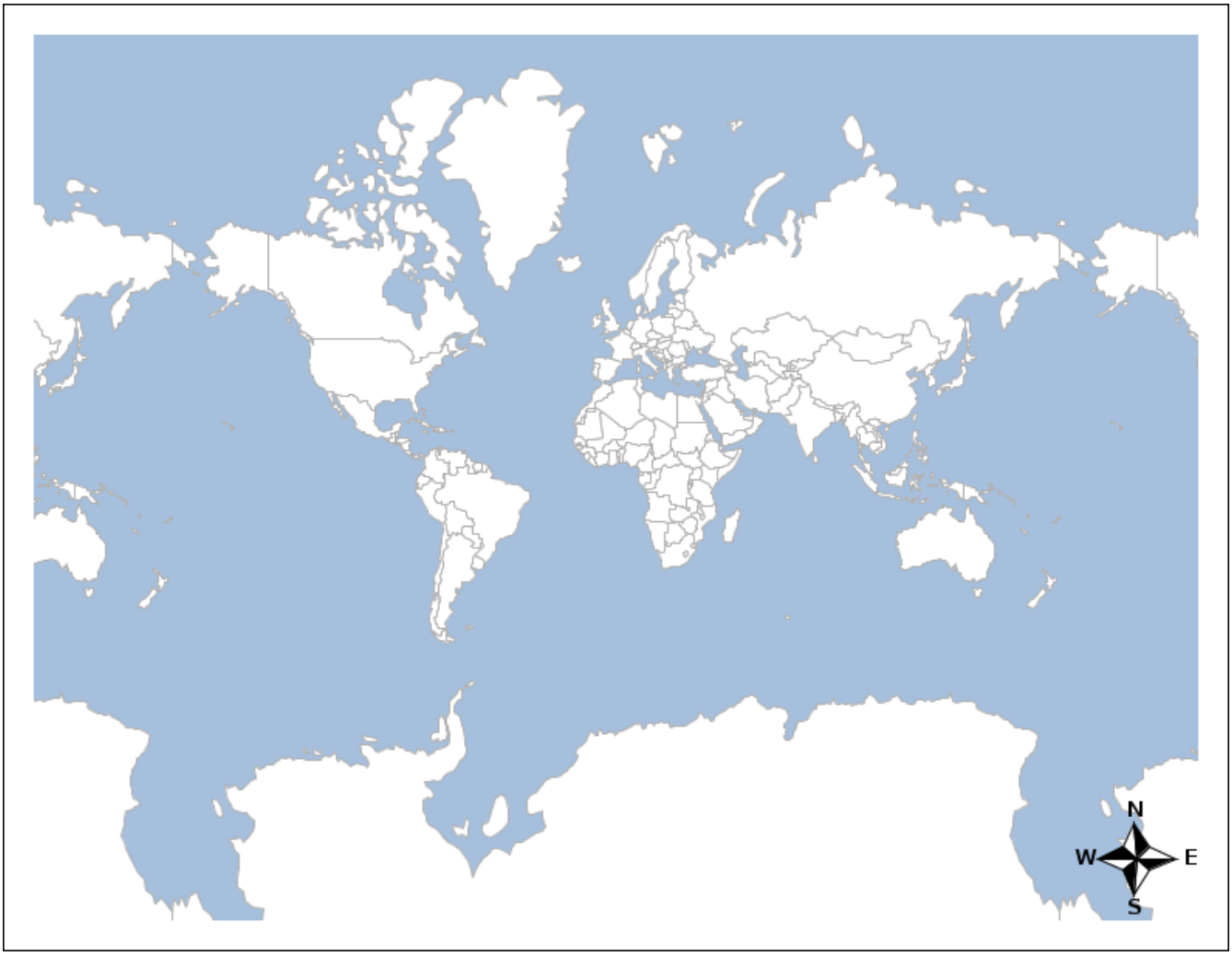
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                    .fillColor(Color.WHITE)
            )
            .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 40).map
(map))
            .northArrow(new NorthArrowItem(pageSize.width - 100, pageSize.height -
100, 80, 80)
                .style(NorthArrowStyle.NorthEastSouthWest)
                .font(new Font("Arial", Font.BOLD, 14))
                .drawText(true))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
fillColor1	java.awt.Color	The first Fill Color
strokeColor1	java.awt.Color	The first Stroke Color
fillColor2	java.awt.Color	The second Fill Color
strokeColor2	java.awt.Color	The second Stroke Color
strokeWidth	float	The width of the stroke
drawText	boolean	Whether to draw text (n,s,e,w) or not
font	java.awt.Font	The Font for the text
textColor	java.awt.Color	The text Color
style	NorthArrowStyle	The North Arrow style (North or NorthEastSouthWest)

Adding a Legend

Add a legend for a Map

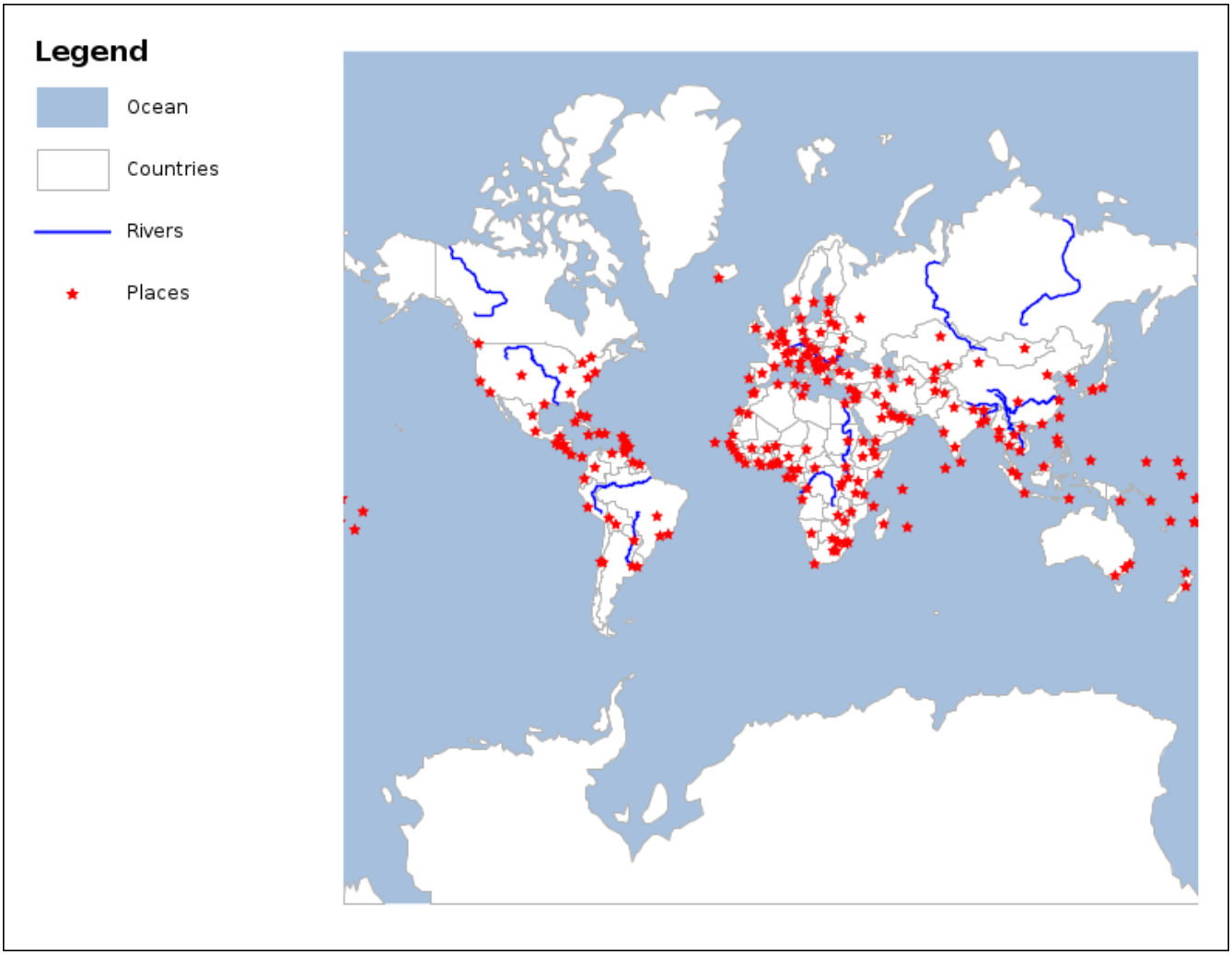
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Layer places = workspace.get("places")
places.style = new Shape("red", 8, "star")
Layer rivers = workspace.get("rivers")
rivers.style = new Stroke("blue", 1)
Map map = new Map(
        layers: [ocean, countries, rivers, places],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                    .fillColor(Color.WHITE)
            )
            .map(new MapItem(220, 20, pageSize.width - 240, pageSize.height - 40).map
(map))
            .legend(new LegendItem(20, 20, 200, pageSize.height - 40).addMap(map))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
backgroudColor	java.awt.Color	The background Color
title	String	The legend title
titleFont	java.awt.Font	The title Font
titleColor	java.awt.Color	The title Color
textFont	java.awt.Font	The text Font
textColor	java.awt.Color	The text Color
entries	List of LegendEntry instances	Legend entries
legendEntryWidth	int	The width of individual entries
legendEntryHeight	int	The height of individual entries
gapBetweenEntries	int	The gap between entries
numberFormat	String	The number format (#.##)

Adding a Date

Add a date

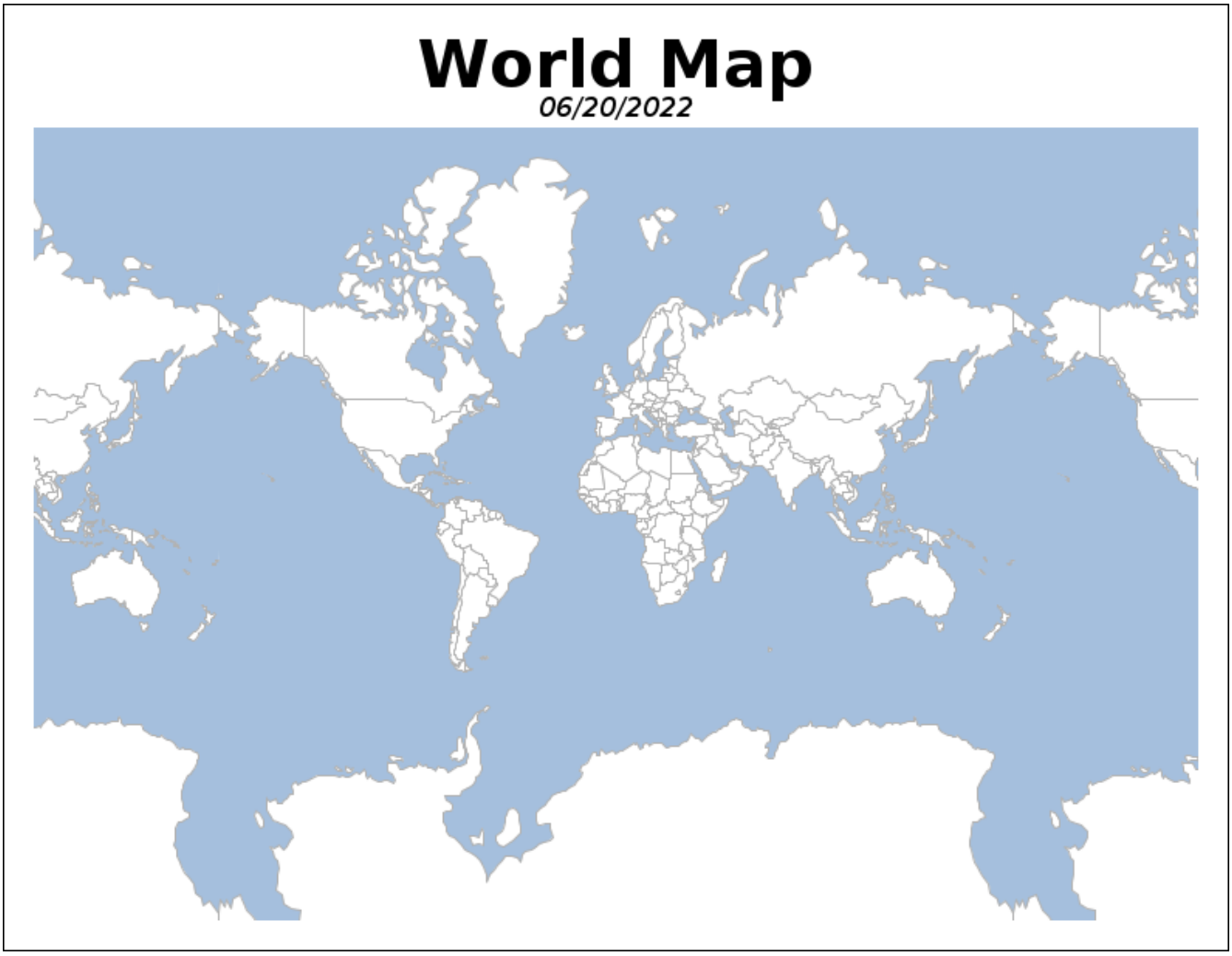
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
        .create(pageSize)
        .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height - 1)
            .fillColor(Color.WHITE)
        )
        .text(new TextItem(20,15, pageSize.width - 40, 60)
            .text("World Map")
            .font(new Font("Arial", Font.BOLD, 42))
            .verticalAlign(VerticalAlign.TOP)
            .horizontalAlign(HorizontalAlign.CENTER)
        )
        .dateText(new DateTextItem(20,58, pageSize.width - 40, 20)
            .font(new Font("Arial", Font.ITALIC, 18))
            .verticalAlign(VerticalAlign.BOTTOM)
            .horizontalAlign(HorizontalAlign.CENTER)
        )
        .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map(map))
        .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
format	String	The date format (MM/dd/yyyy)
date	Date	The Date to display
color	java.awt.Color	The text Color
font	java.awt.Font	The text Font
horizontalAlign	HorizontalAlign	The horizontal alingment of the text
verticalAlign	VerticalAlign	The vertical alingment of the text

Adding Scale Text

Add scale text

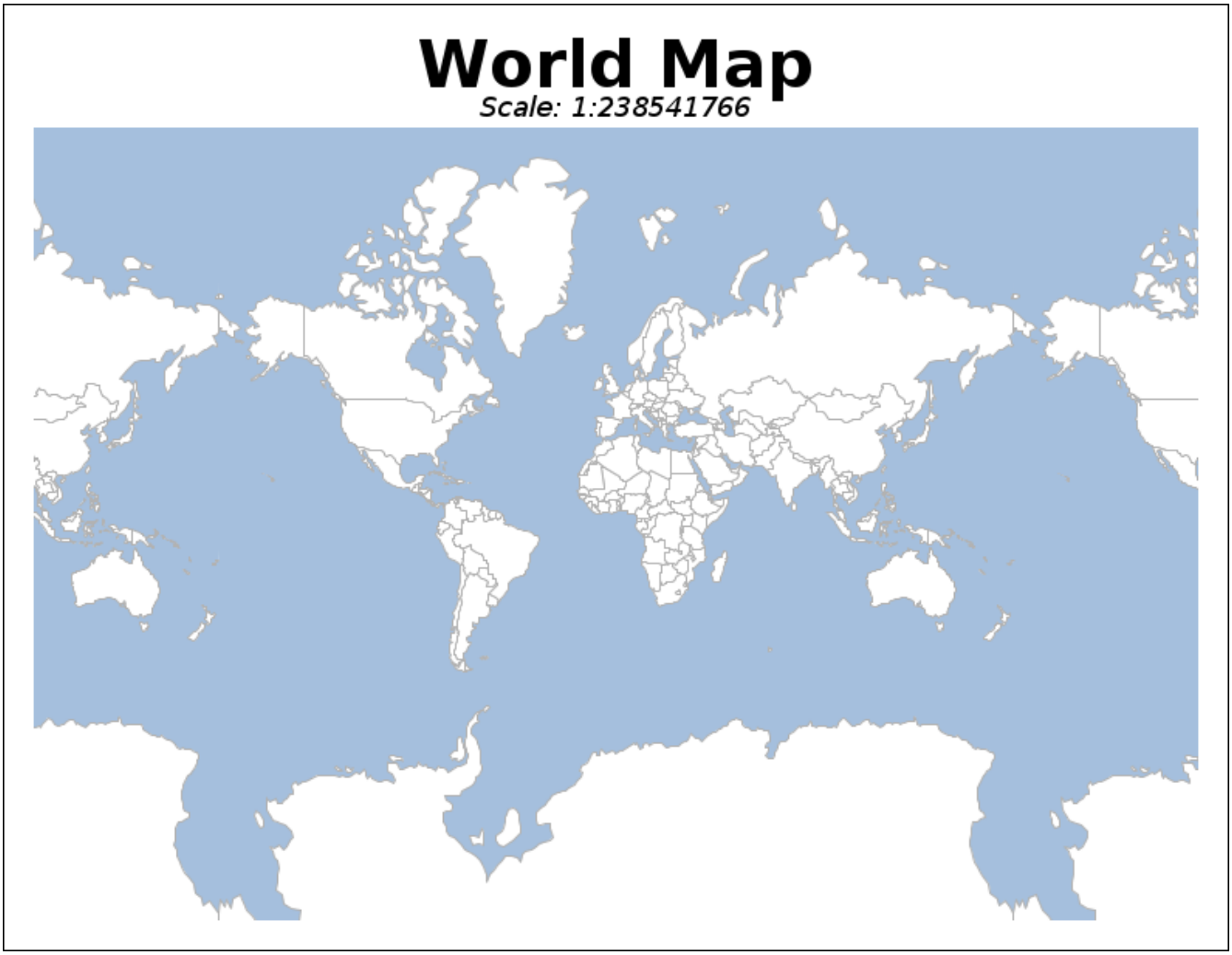
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
        .create(pageSize)
        .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height - 1)
            .fillColor(Color.WHITE)
        )
        .text(new TextItem(20,15, pageSize.width - 40, 60)
            .text("World Map")
            .font(new Font("Arial", Font.BOLD, 42))
            .verticalAlign(VerticalAlign.TOP)
            .horizontalAlign(HorizontalAlign.CENTER)
        )
        .scaleText(new ScaleTextItem(20,58, pageSize.width - 40, 20)
            .map(map)
            .format("#")
            .prefixText("Scale: ")
            .font(new Font("Arial", Font.ITALIC, 18))
            .verticalAlign(VerticalAlign.BOTTOM)
            .horizontalAlign(HorizontalAlign.CENTER)
        )
        .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map(map))
        .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
map	geoscript.render.Map	The Map to get the scale from
color	java.awt.Color	The text Color
font	java.awt.Font	The text Font
horizontalAlign	HorizontalAlign	The horizontal alingment of the text
verticalAlign	VerticalAlign	The vertical alingment of the text
format	String	The number format for displaying the scale
prefixText	String	The text to display before the scale (Scale: ...)

Adding Scale Bar

Add scale bar

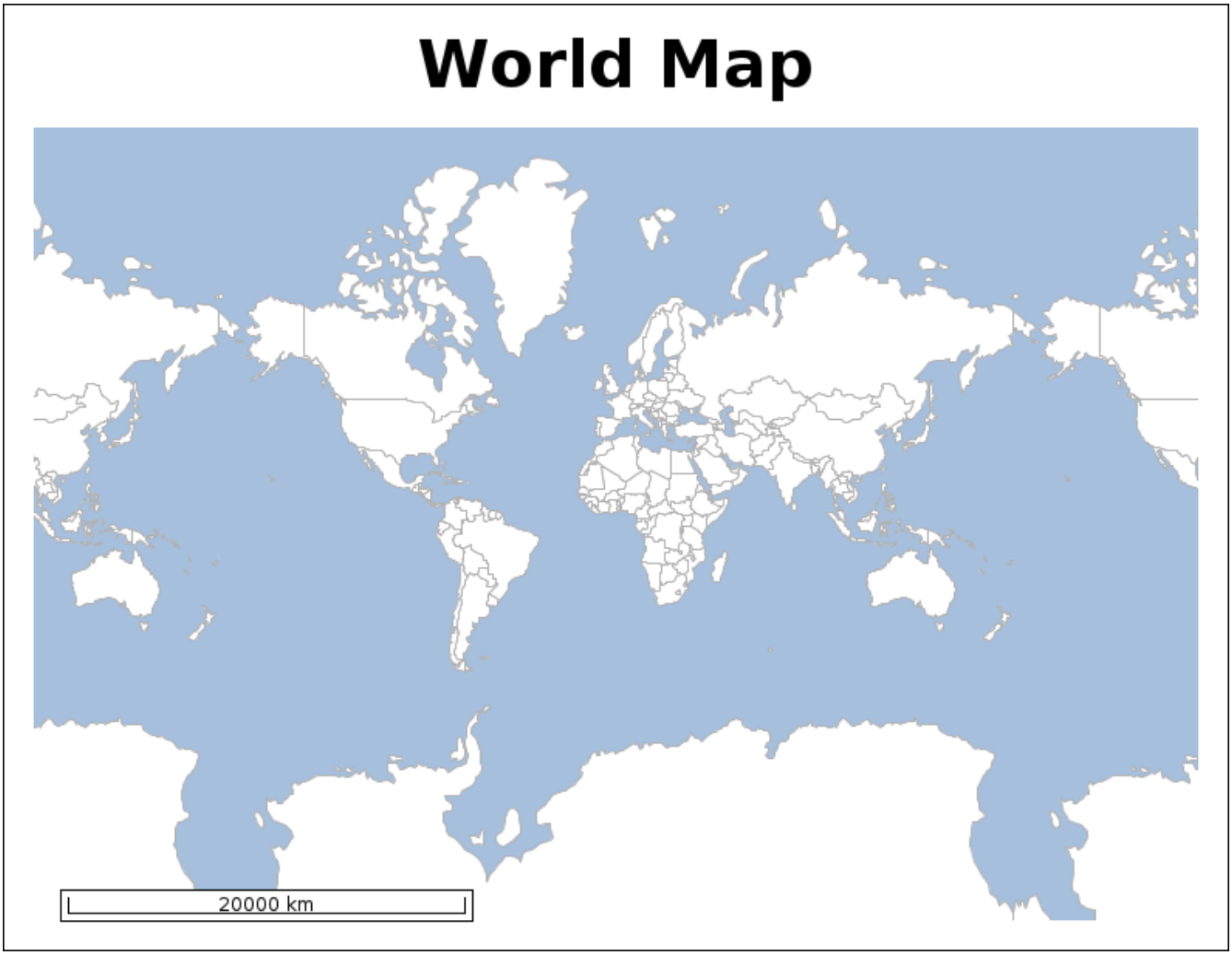
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                    .fillColor(Color.WHITE)
            )
            .text(new TextItem(20,15, pageSize.width - 40, 60)
                    .text("World Map")
                    .font(new Font("Arial", Font.BOLD, 42))
                    .verticalAlign(VerticalAlign.TOP)
                    .horizontalAlign(HorizontalAlign.CENTER)
            )
            .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map
(map))
            .scaleBar(new ScaleBarItem(20,pageSize.height - 40, 300, 20)
                    .map(map)
                    .units(ScaleBarItem.Units.METRIC)
            )
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
map	geoscript.render.Map	The Map to get the scale from
strokeColor	java.awt.Color	The stroke Color
fillColor	java.awt.Color	The fill Color
strokeWidth	float	The stroke width
font	java.awt.Font	The text Font
border	int	The border padding
units	Units	The units (metric or us)

Adding a Line

Add a line

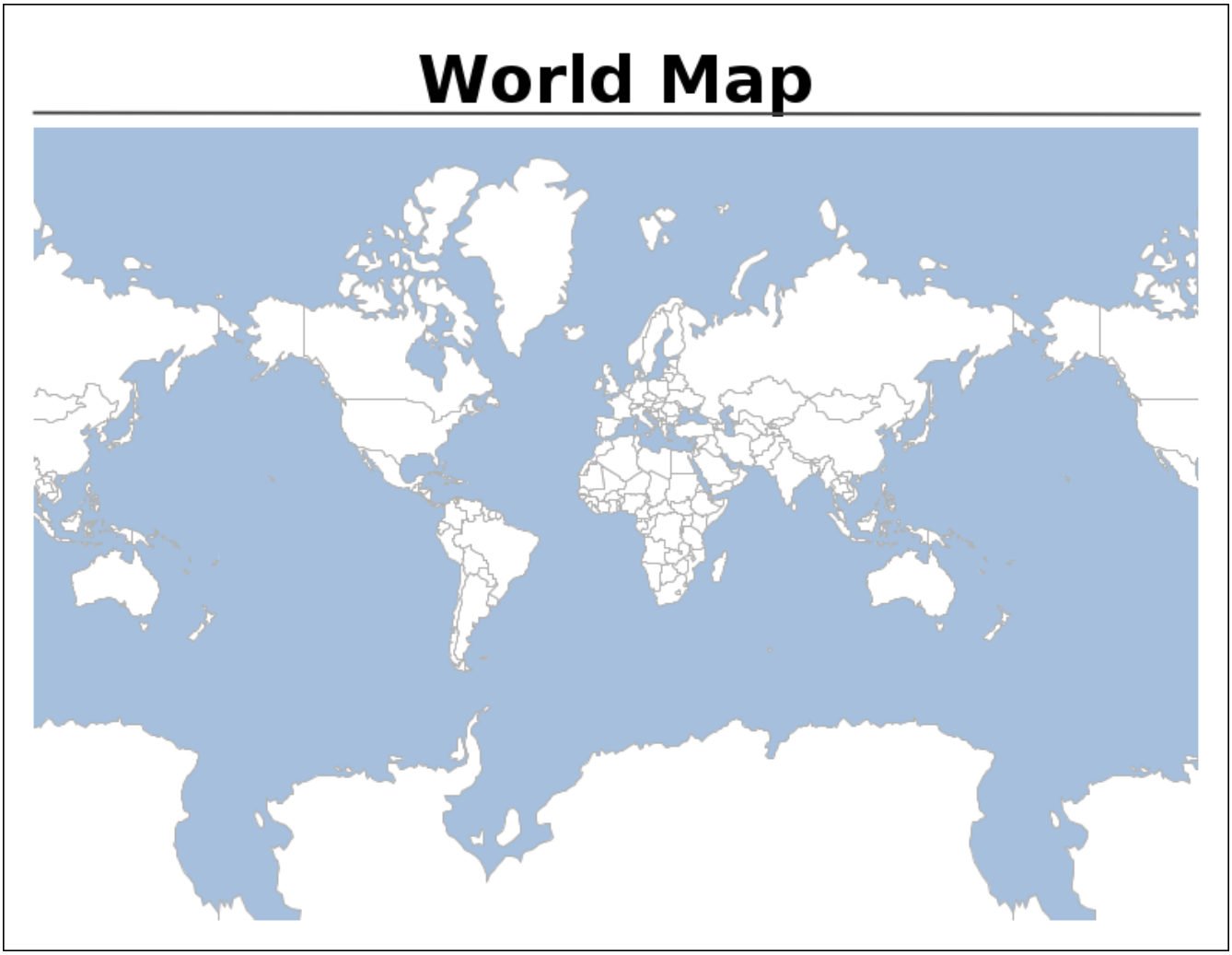
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .text(new TextItem(20,20, pageSize.width - 40, 60)
                .text("World Map")
                .font(new Font("Arial", Font.BOLD, 42))
                .verticalAlign(VerticalAlign.MIDDLE)
                .horizontalAlign(HorizontalAlign.CENTER)
            )
            .line(new LineItem(20, 70, pageSize.width - 40, 1)
                .strokeWidth(2)
                .strokeColor(Color.DARK_GRAY)
            )
            .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map
(map))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
map	geoscript.render.Map	The Map to get the scale from
strokeColor	java.awt.Color	The stroke Color
strokeWidth	float	The stroke width

Adding a Grid

Adding a grid is a nice way to placing items.

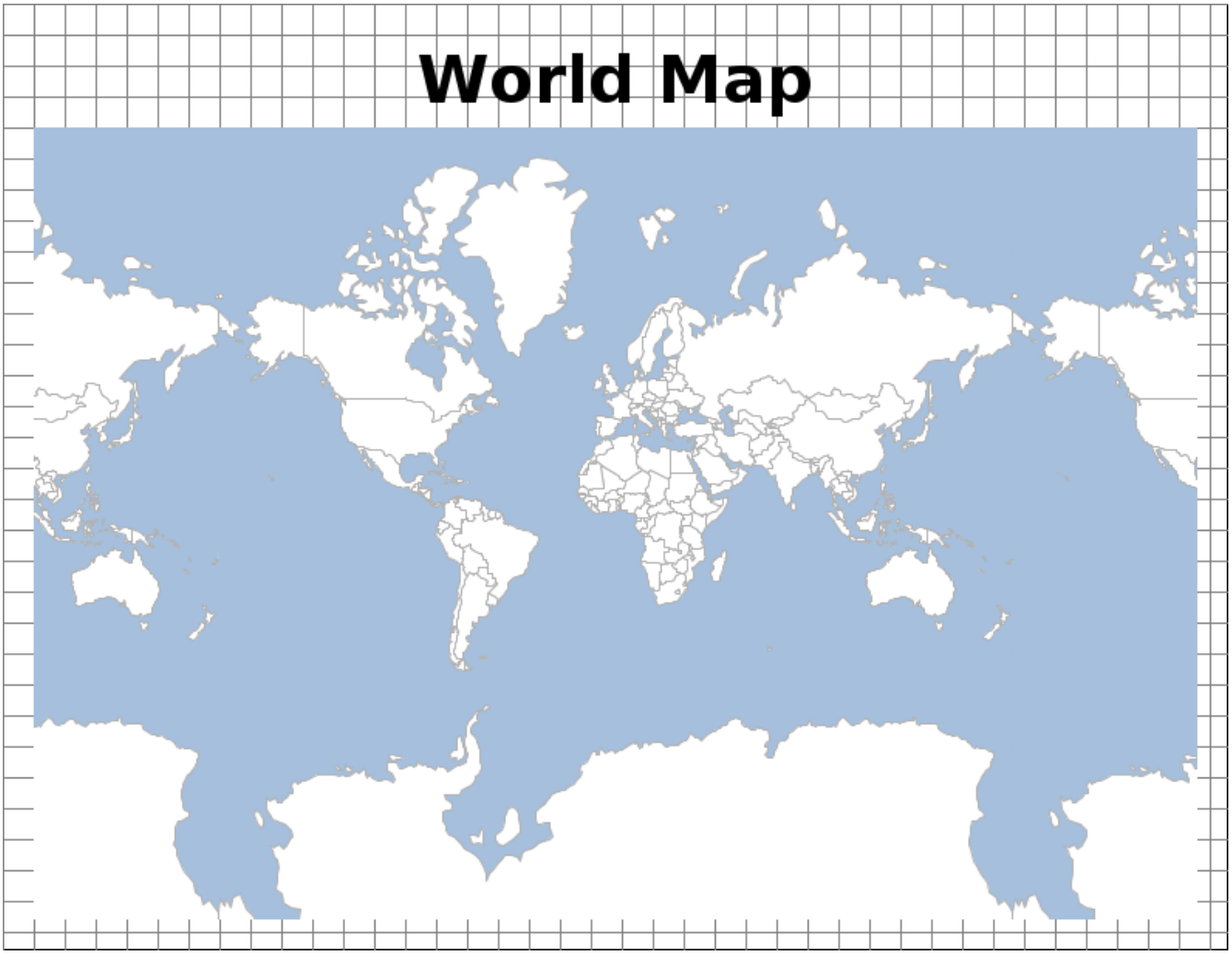
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .grid(new GridItem(0,0,pageSize.width, pageSize.height)
                .size(20)
                .strokeColor(Color.GRAY)
                .strokeWidth(1.0)
            )
            .text(new TextItem(20,20, pageSize.width - 40, 60)
                .text("World Map")
                .font(new Font("Arial", Font.BOLD, 42))
                .verticalAlign(VerticalAlign.MIDDLE)
                .horizontalAlign(HorizontalAlign.CENTER)
            )
            .map(new MapItem(20, 80, pageSize.width - 40, pageSize.height - 100).map
(map))
            .build(outputStream)

}
```



Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
size	int	The cell size
strokeColor	java.awt.Color	The stroke Color
strokeWidth	float	The stroke width

Adding a Paragraph

Adding a paragraph of text

```
        Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
        Layer countries = workspace.get("countries")
        countries.style = new SLDReader().read(new File
('src/main/resources/countries.sld'))
        Layer ocean = workspace.get("ocean")
        ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
        Map map = new Map(
                layers: [ocean, countries],
                bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject(
"EPSG:3857"),
                projection: new Projection("EPSG:3857")
        )

        File file = new File("map.png")
        file.withOutputStream { OutputStream outputStream ->

            PageSize pageSize = PageSize.LETTER_LANDSCAPE

            CartoFactories.findByName("png")
                .create(pageSize)
                .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height
- 1)
                    .fillColor(Color.WHITE)
                )
                .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 100
).map(map))
                .paragraph(new ParagraphItem(20, pageSize.height - 60, pageSize.width
- 40, 60)
                    .font(new Font("Arial", Font.PLAIN, 12))
                    .color(Color.BLACK)
                    .text("""Natural Earth is a public domain map dataset available at
1:10m, 1:50m, and 1:110 million scales.
Featuring tightly integrated vector and raster data, with Natural Earth you can make a
variety of visually pleasing,
well-crafted maps with cartography or GIS software.
""")
                )
                .build(outputStream)

        }
```

Natural Earth is a public domain map dataset available at 1:10m, 1:50m, and 1:110 million scales. Featuring tightly integrated vector and raster data, with Natural Earth you can make a variety of visually pleasing, well-crafted maps with cartography or GIS software.

Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
text	String	The paragraph text
font	java.awt.Font	The text Font
color	java.awt.Color	The text Color

Adding an Image

Adding an image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries],
        bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                .fillColor(Color.WHITE)
            )
            .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 40).map
(map))
            .image(new ImageItem(pageSize.width - 100, pageSize.height - 100, 80, 80)
                .path(new File("src/main/resources/image.png"))
            )
            .build(outputStream)

}
```

Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
path	File or URL	The source path of the image

Adding a Table

Adding a Table

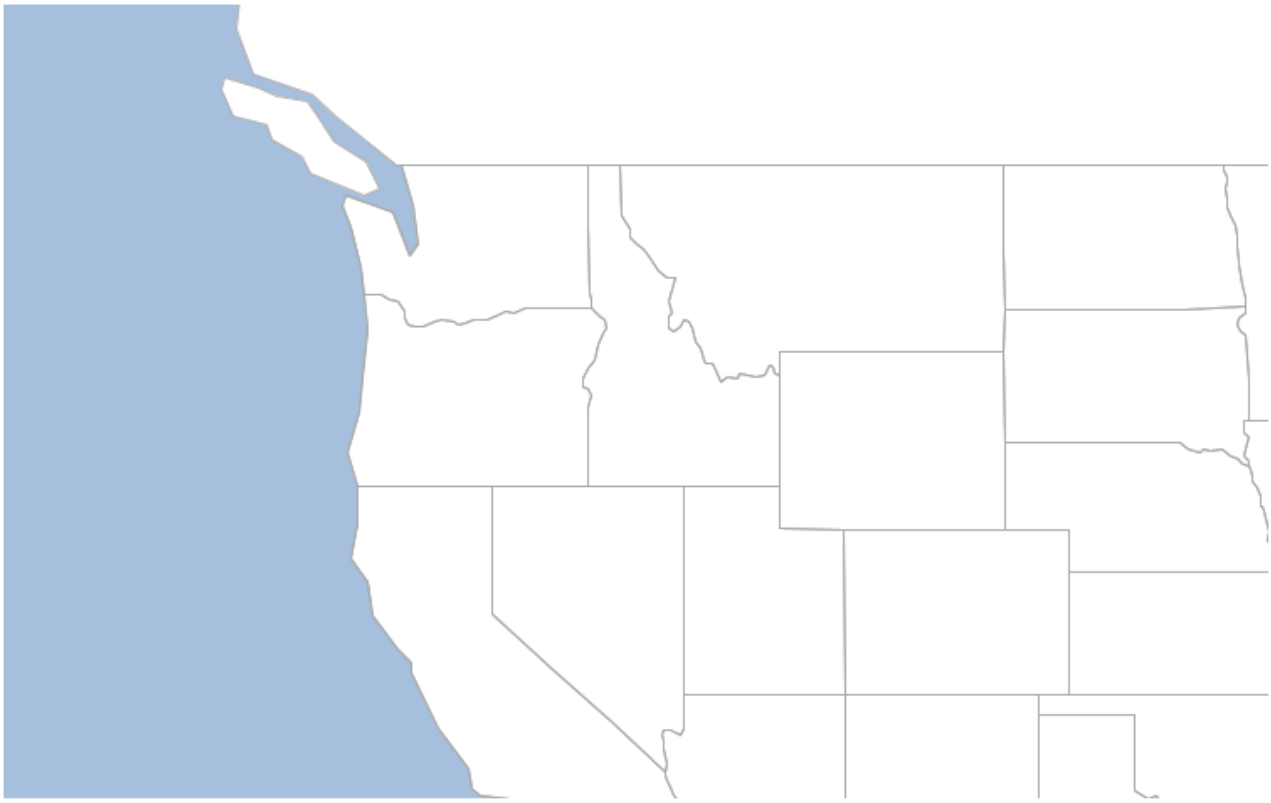
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer states = workspace.get("states")
states.style = new SLDReader().read(new File('src/main/resources/states.sld'))
Layer countries = workspace.get("countries")
countries.style = new SLDReader().read(new File('src/main/resources/countries.sld'))
Layer ocean = workspace.get("ocean")
ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
Map map = new Map(
        layers: [ocean, countries, states],
        bounds: new Bounds(-135.225466,36.256870,-95.850466,50.746340, "EPSG:4326"
).reproject("EPSG:3857"),
        projection: new Projection("EPSG:3857")
)

File file = new File("map.png")
file.withOutputStream { OutputStream outputStream ->

    PageSize pageSize = PageSize.LETTER_LANDSCAPE

    CartoFactories.findByName("png")
            .create(pageSize)
            .rectangle(new RectangleItem(0, 0, pageSize.width - 1, pageSize.height -
1)
                    .fillColor(Color.WHITE)
            )
            .map(new MapItem(20, 20, pageSize.width - 40, pageSize.height - 140).map
(map))
            .table(new TableItem(20, pageSize.height - 100, pageSize.width - 40, 80)
                .columns(["Name", "Abbreviation"])
                .row([[Name: "Washington", Abbreviation: "WA"]])
                .row([[Name: "Oregon", Abbreviation: "OR"]])
                .row([[Name: "California", Abbreviation: "CA"]])
            )
            .build(outputStream)

}
```



Name	Abbreviation
Washington	WA
Oregon	OR
California	CA

Property	Type	Description
x	int	The number of pixels from left
y	int	The number of pixels from top
width	int	The width of the item in pixels
height	int	The height of the item in pixels
columns	List of Strings	The column names
row	A Maps of values	One row of values where the keys are the column names.
rows	A List of Maps	The data as a List of Maps where the keys are the column names.
columnRowStyle	TableItem.RowStyle	backGroudColor, font, textColor, strokeColor for columns
evenRowStyle	TableItem.RowStyle	backGroudColor, font, textColor, strokeColor for even rows

Property	Type	Description
oddRowStyle	TableItem.RowStyle	backGroudColor, font, textColor, strokeColor for odd rows

Builders

CartoBuilders write cartographic documents to different formats. Images, PDFs, and SVGs are currently supported.

ImageCartoBuilder

The ImageCartoBuilder can produce PNG or JPEG images.

Use the ImageCartoBuilder to create a PNG image.

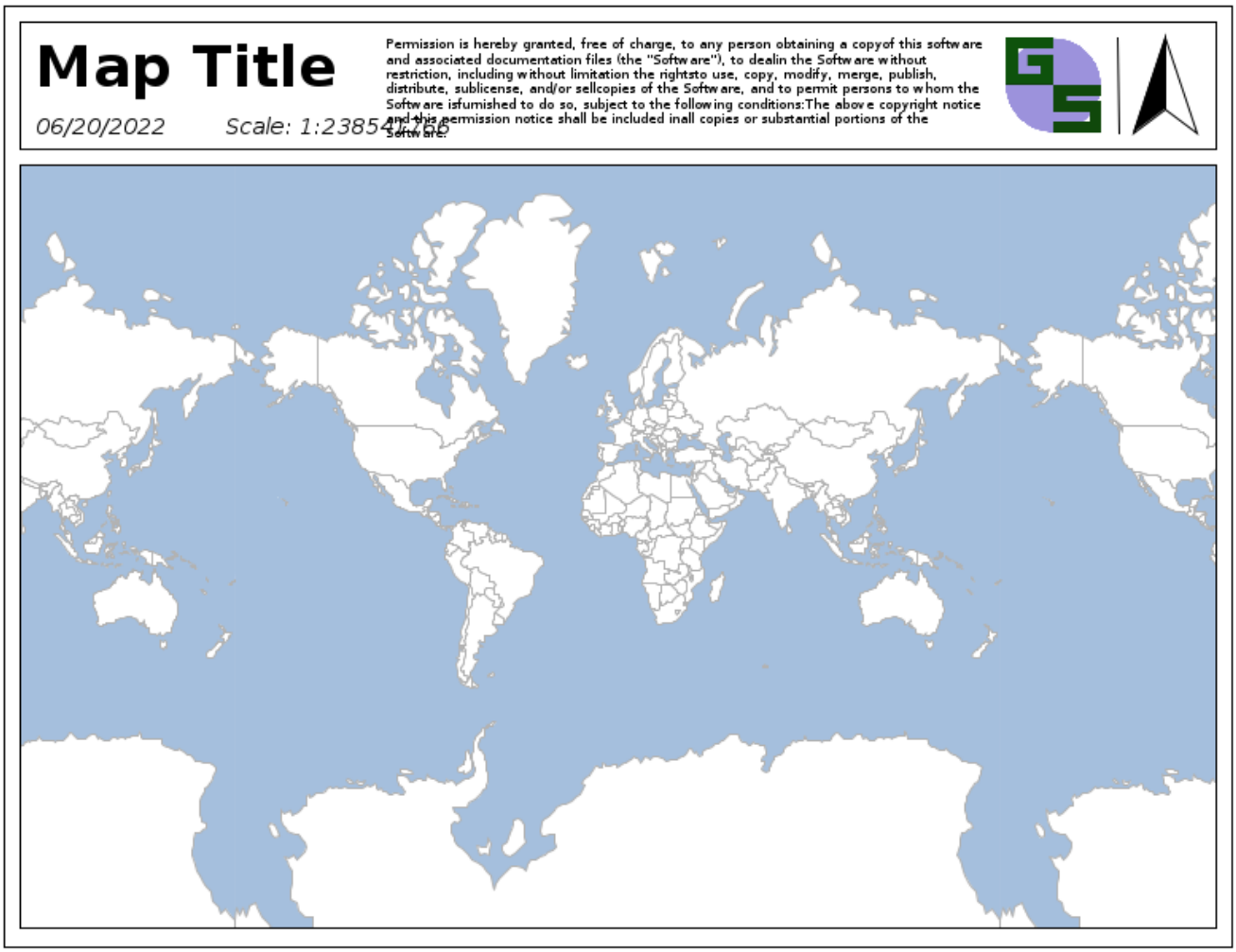
```
        Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
        Layer countries = workspace.get("countries")
        countries.style = new SLDReader().read(new File
('src/main/resources/countries.sld'))
        Layer ocean = workspace.get("ocean")
        ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
        Map map = new Map(
                layers: [ocean, countries],
                bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject(
"EPSG:3857"),
                projection: new Projection("EPSG:3857")
        )

        File file = new File("map.png")
        file.withOutputStream { OutputStream outputStream ->

            new ImageCartoBuilder(PageSize.LETTER_LANDSCAPE, ImageCartoBuilder
.ImageType.PNG)
                .rectangle(new RectangleItem(0, 0, 792, 612).strokeColor(Color.WHITE
).fillColor(Color.WHITE))
                .rectangle(new RectangleItem(10, 10, 772, 592))
                .rectangle(new RectangleItem(20, 20, 752, 80))
                .text(new TextItem(30, 50, 200, 20).text("Map Title").font(new Font
("Arial", Font.BOLD, 36)))
                .dateText(new DateTextItem(30, 85, 200, 10).font(new Font("Arial",
Font.ITALIC, 14)))
                .scaleText(new ScaleTextItem(150, 85, 200, 10).map(map).font(new Font
("Arial", Font.ITALIC, 14)))
                .paragraph(new ParagraphItem(250, 30, 380, 70).text("""Permission is
hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
""").font(new Font("Arial", Font.PLAIN, 8)))
                .line(new LineItem(710, 30, 1, 60))
                .image(new ImageItem(640, 30, 60, 60).path(new File(getClass
().getClassLoader().getResource("image.png").toURI())))
                .northArrow(new NorthArrowItem(720, 30, 40, 60))
                .map(new MapItem(20, 110, 752, 480).map(map))
                .rectangle(new RectangleItem(20, 110, 752, 480))

                .build(outputStream)

        }
```



PdfCartoBuilder

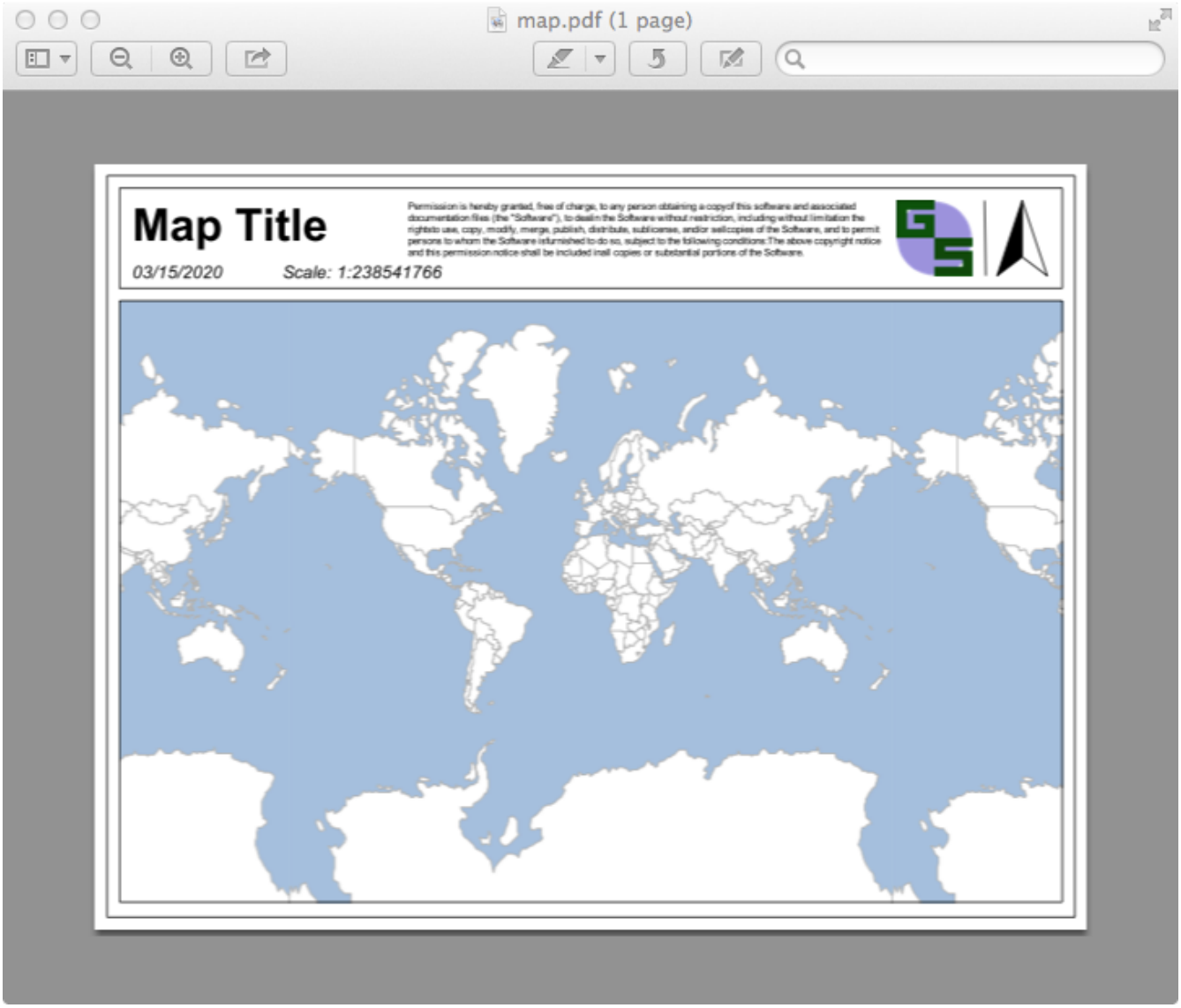
Use the PdfCartoBuilder to create a PDF.

```
        Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
        Layer countries = workspace.get("countries")
        countries.style = new SLDReader().read(new File
('src/main/resources/countries.sld'))
        Layer ocean = workspace.get("ocean")
        ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
        Map map = new Map(
                layers: [ocean, countries],
                bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject(
"EPSG:3857"),
                projection: new Projection("EPSG:3857")
        )

        File file = new File("map.pdf")
        file.withOutputStream { OutputStream outputStream ->

            new PdfCartoBuilder(PageSize.LETTER_LANDSCAPE)
                .rectangle(new RectangleItem(0, 0, 792, 612).strokeColor(Color.WHITE
).fillColor(Color.WHITE))
                .rectangle(new RectangleItem(10, 10, 772, 592))
                .rectangle(new RectangleItem(20, 20, 752, 80))
                .text(new TextItem(30, 50, 200, 20).text("Map Title").font(new Font
("Arial", Font.BOLD, 36)))
                .dateText(new DateTextItem(30, 85, 200, 10).font(new Font("Arial",
Font.ITALIC, 14)))
                .scaleText(new ScaleTextItem(150, 85, 200, 10).map(map).font(new Font
("Arial", Font.ITALIC, 14)))
                .paragraph(new ParagraphItem(250, 30, 380, 70).text("""Permission is
hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
""").font(new Font("Arial", Font.PLAIN, 8)))
                .line(new LineItem(710, 30, 1, 60))
                .image(new ImageItem(640, 30, 60, 60).path(new File(getClass
().getClassLoader().getResource("image.png").toURI())))
                .northArrow(new NorthArrowItem(720, 30, 40, 60))
                .map(new MapItem(20, 110, 752, 480).map(map))
                .rectangle(new RectangleItem(20, 110, 752, 480))
                .build(outputStream)

        }
```



SvgCartoBuilder

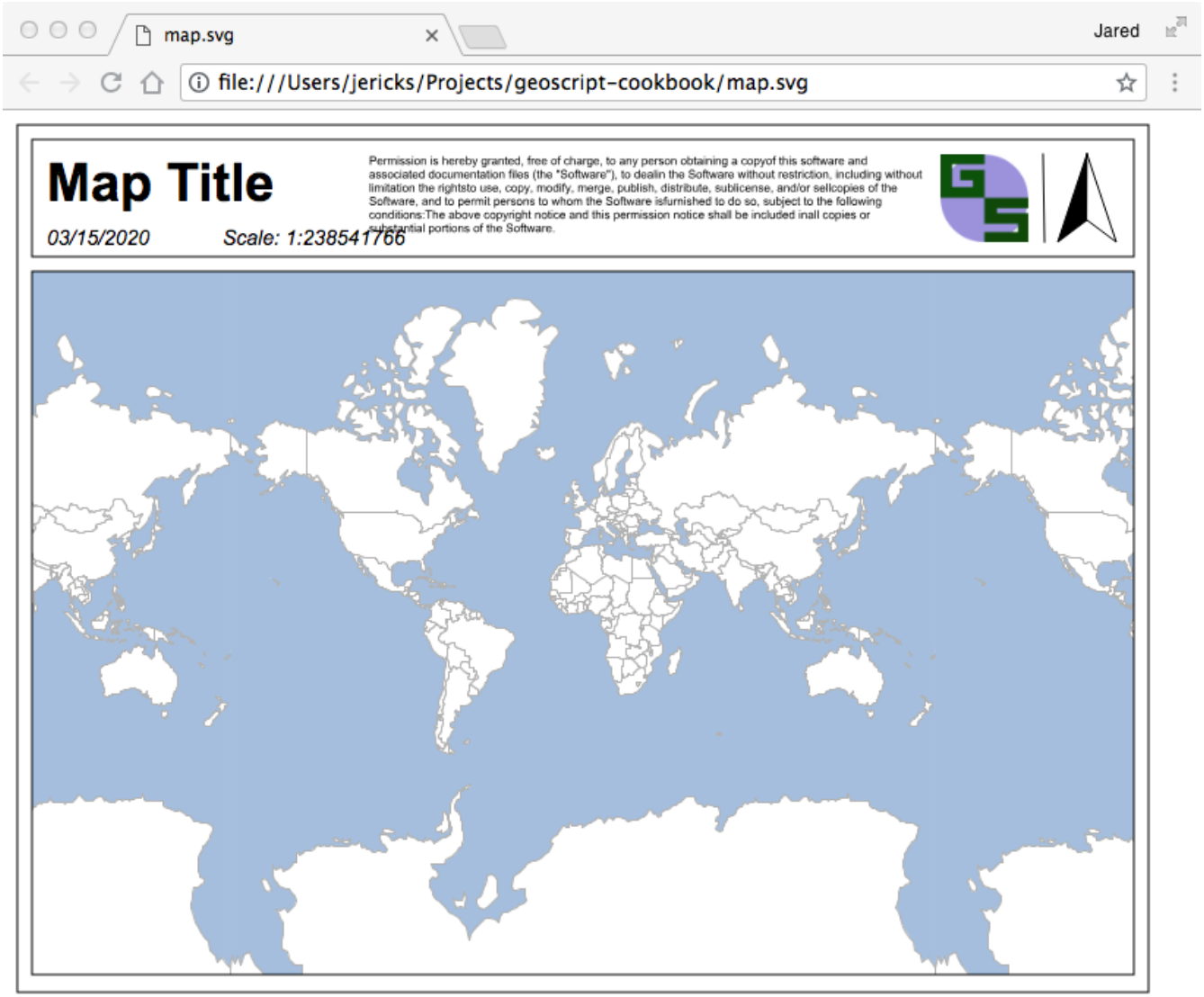
Use the SvgCartoBuilder to create a SVG document.

```
        Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
        Layer countries = workspace.get("countries")
        countries.style = new SLDReader().read(new File
('src/main/resources/countries.sld'))
        Layer ocean = workspace.get("ocean")
        ocean.style = new SLDReader().read(new File('src/main/resources/ocean.sld'))
        Map map = new Map(
                layers: [ocean, countries],
                bounds: new Bounds(-180,-85,180,85, "EPSG:4326").reproject(
"EPSG:3857"),
                projection: new Projection("EPSG:3857")
        )

        File file = new File("map.svg")
        file.withOutputStream { OutputStream outputStream ->

            new SvgCartoBuilder(PageSize.LETTER_LANDSCAPE)
                .rectangle(new RectangleItem(0, 0, 792, 612).strokeColor(Color.WHITE
).fillColor(Color.WHITE))
                .rectangle(new RectangleItem(10, 10, 772, 592))
                .rectangle(new RectangleItem(20, 20, 752, 80))
                .text(new TextItem(30, 50, 200, 20).text("Map Title").font(new Font
("Arial", Font.BOLD, 36)))
                .dateText(new DateTextItem(30, 85, 200, 10).font(new Font("Arial",
Font.ITALIC, 14)))
                .scaleText(new ScaleTextItem(150, 85, 200, 10).map(map).font(new Font
("Arial", Font.ITALIC, 14)))
                .paragraph(new ParagraphItem(250, 30, 380, 70).text("""Permission is
hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
""").font(new Font("Arial", Font.PLAIN, 8)))
                .line(new LineItem(710, 30, 1, 60))
                .image(new ImageItem(640, 30, 60, 60).path(new File(getClass
().getClassLoader().getResource("image.png").toURI())))
                .northArrow(new NorthArrowItem(720, 30, 40, 60))
                .map(new MapItem(20, 110, 752, 480).map(map))
                .rectangle(new RectangleItem(20, 110, 752, 480))
                .build(outputStream)

        }
```



Reading CartoBuilders

The IO module can read a CartoBuilder from JSON or XML documents.

Finding Carto Readers

List all Carto Readers

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

```
json
xml
```

Find a Carto Reader

```
CartoReader reader = CartoReaders.find("json")
println reader.name
```

```
json
```

JSON

Read a CartoBuilder from a JSON String

```
        String json = """{
    "type": "png",
    "width": 400,
    "height": 400,
    "items": [
        {
            "x": 0,
            "y": 0,
            "width": 400,
            "height": 400,
            "type": "rectangle",
            "fillColor": "white",
            "strokeColor": "white"
        },
        {
            "x": 10,
            "y": 10,
            "width": 380,
            "height": 380,
            "type": "rectangle"
        },
        {
            "x": 20,
            "y": 20,
            "width": 360,
            "height": 360,
            "type": "map",
            "name": "mainMap",
            "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"
            }
        ]
    },
    {
        "x": 20,
        "y": 20,
        "width": 30,
        "height": 40,
        "type": "northarrow"
    },
    {
        "x": 260,
        "y": 20,
        "width": 50,
        "height": 200,
        "type": "legend",
        "map": "mainMap"
    },
    {
        "x": 70,
        "y": 20,
        "width": 170,
        "height": 50,
        "type": "text",
        "text": "Western US",
        "font": {
            "name": "Arial",
            "style": "BOLD",
            "size": 24
        },
        "horizontalAlign": "CENTER",
        "verticalAlign": "MIDDLE"
    }
```

```
    ]
}
"""
        CartoReader cartoReader = new JsonCartoReader()
        CartoBuilder cartoBuilder = cartoReader.read(json)
        File file = new File("target/carto_from_json.png")
        file.withOutputStream { OutputStream outputStream ->
            cartoBuilder.build(outputStream)
        }
```

XML

Read a CartoBuilder from an XML String

```
        String json = """<carto>
    <type>png</type>
    <width>400</width>
    <height>400</height>
    <items>
        <item>
            <x>0</x>
            <y>0</y>
            <width>400</width>
            <height>400</height>
            <type>rectangle</type>
            <fillColor>white</fillColor>
            <strokeColor>white</strokeColor>
        </item>
        <item>
```

```
            <x>10</x>
            <y>10</y>
            <width>380</width>
            <height>380</height>
            <type>rectangle</type>
        </item>
        <item>
            <x>20</x>
            <y>20</y>
            <width>360</width>
            <height>360</height>
            <type>map</type>
            <name>mainMap</name>
            <proj>EPSG:4326</proj>
            <bounds>
                <minX>-135.911779</minX>
                <minY>36.993573</minY>
                <maxX>-96.536779</maxX>
                <maxY>51.40589</maxY>
            </bounds>
            <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>
        </item>
        <item>
            <x>20</x>
            <y>20</y>
            <width>30</width>
            <height>40</height>
            <type>northarrow</type>
        </item>
```

```
        <item>
            <x>260</x>
            <y>20</y>
            <width>50</width>
            <height>200</height>
            <type>legend</type>
            <map>mainMap</map>
        </item>
        <item>
            <x>70</x>
            <y>20</y>
            <width>170</width>
            <height>50</height>
            <type>text</type>
            <text>Western US</text>
            <font>
                <name>Arial</name>
                <style>BOLD</style>
                <size>24</size>
            </font>
            <horizontalAlign>CENTER</horizontalAlign>
            <verticalAlign>MIDDLE</verticalAlign>
        </item>
    </items>
</carto>
"""
        CartoReader cartoReader = new XmlCartoReader()
        CartoBuilder cartoBuilder = cartoReader.read(json)
        File file = new File("target/carto_from_xml.png")
        file.withOutputStream { OutputStream outputStream ->
            cartoBuilder.build(outputStream)
        }
```


Western US
Legend
Ocean
Countries
States