

STYLING WITH GEOSERVER

This tutorial covers introductory SLD styling in GeoServer for points, lines, polygons, and rasters utilizing base XML from the SLD Cookbook.



GeoServer

James Raines

University of Colorado Denver

Spring 2017

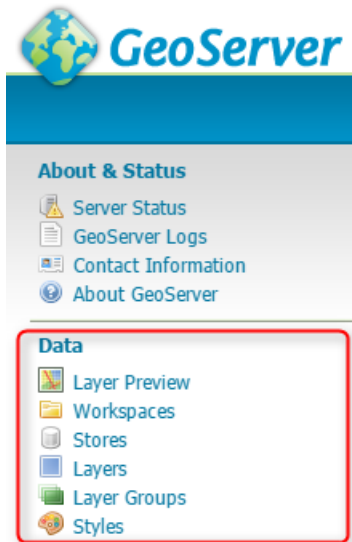


COLLEGE OF LIBERAL ARTS AND SCIENCES
Geography and Environmental Sciences
UNIVERSITY OF COLORADO DENVER

Login to GeoServer:

1. Login to a VPN service such as Cisco
2. Connect to anywhere.ucdenver.edu and login with your student credentials
3. Open Remote Desktop Connection and connect to:
geoserver.ucdenver.pvt
4. Login with your student credentials
5. In a Firefox or Chrome browser, (do not use Internet Explorer)
navigate to <http://geoserver.ucdenver.pvt:8181/geoserver/web>
6. Login with your GeoServer credentials

***For the purposes of this tutorial, we will only be working in the Data section:



Create New Data Stores

In this step, we will be creating new data stores to host the following layers:

- DenverBoundary.shp
- DenverArterials.shp
- SpeedRecordings.shp
- VehCrashes_HeatMap.tif

By this time, you should already have a workspace. In your workspace, create two new data stores that link to the GeoServer_Styling folder.

- 1) A Shapefile data store: your *Shapefile location* should appear like this "file:data_dir/GeoServer_Styling". This will enable you to publish all the shapefiles inside of this folder.
- 2) GeoTIFF data store: GeoTIFF data stores can only publish one GeoTIFF at a time, so your *URL** should appear like this "file:data_dir/GeoServer_Styling/VehCrashes_HeatMap.tif"

Publish Data

Publish all 3 shapefiles and the 1 raster.

- 1) When publishing shapefiles:
 - a. Use EPSG:2773 for: DenverBoundary, DenverArterials, and SpeedRecordings

Vector Data Sources

- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and publishes them as a single layer.
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information.
- Shapefile - ESRI(tm) Shapefiles (*.shp)**
- Web Feature Server (NG) - Provides access to the Features published on the server (when supported / allowed).

Raster Data Sources

- ArcGrid - ARC/INFO ASCII GRID Coverage Format
- GeoTIFF - Tagged Image File Format with Geographic information**
- Gtopo30 - Gtopo30 Coverage Format
- ImageMosaic - Image mosaicking plugin
- WorldImage - A raster file accompanied by a spatial data file

Coordinate Reference Systems

Native SRS

UNKNOWN

Declared SRS

EPSG:2773

- b. Compute Bounding Boxes from data and native bounds

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
-99.304014999999	29.628089000000	-95.779677999999	32.012525999999

☒ Compute from data

☐ Compute from SRS bounds

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
-103.5187865259	-6.145563423210	-103.5187593204	-6.145544444591

☒ Compute from native bounds

- c. Under the *Publishing* tab, scroll down and ensure that *Queryable* is checked.



WMS Settings

Layer Settings

☒ Queryable

☐ Opaque

- d. Save the layer.



2) When publishing the raster:

- a. Use EPSG:2773
- b. Compute Bounding Boxes from data and native bounds, just like the shapefiles
- c. Navigate to the *Publishing* tab
- d. For *Interpolation Methods*, change bilinear to the default, and ensure that nearest neighbor, bilinear, and bicubic are selected
- e. For *Formats*, ensure that GeoTIFF is the Native Format
- f. Check *Queryable*
- g. Save the layer.

Interpolation Methods

Default Interpolation Method

bilinear ▼

Current Interpolation Methods

Available		Selected
	⇒	nearest neighbor
	⇐	bilinear
		bicubic

Formats

Native Format

GeoTIFF

Supported Formats

Available Formats		Selected Formats
GTOPO30	⇒	GIF
IMAGEMOSAIC	⇐	PNG
ARCGRID		JPEG
		TIFF
		GEOTIFF

Your data stores should now appear as below:

Shapefile:

Published	Layer name	Action
✓	DenverArterials	Publish again
✓	DenverBoundary	Publish again
✓	SpeedRecordings	Publish again

GeoTIFF:

Published	Layer name	Action
✓	VehCrashes_HeatMap	Publish again

SLD Cookbook

Navigate to:

<http://docs.geoserver.org/stable/en/user/styling/sld/cookbook/>

Here, you will find XML base code and sample visualizations for styling points, lines, polygons, and rasters.

Click on a few and get familiar with what is available.

Text Editors

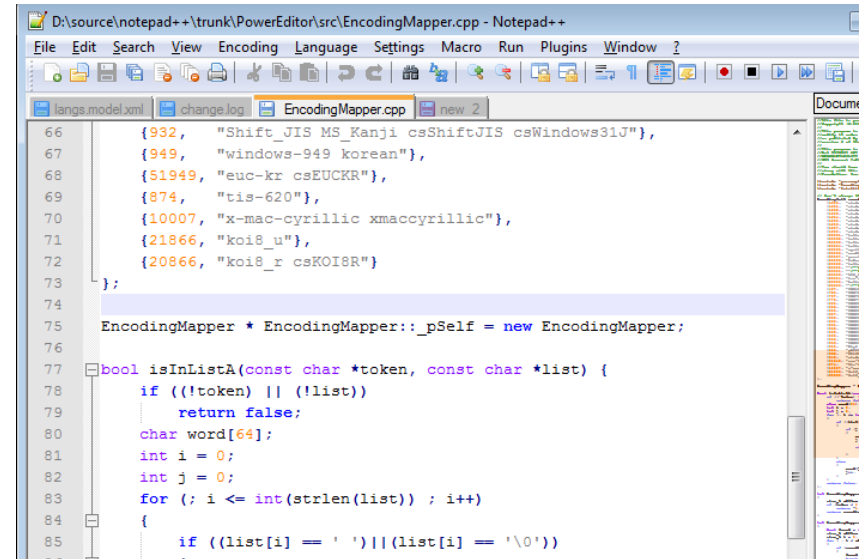
For the rest of this tutorial, we will be editing XML code in a text editor. There are many out there with varying benefits. Notepad++ is one of the most widely used, traditional text editors, but others, including Sublime and Atom, offer more robust functionality.

For this exercise, we will be using Sublime. Reference this document before proceeding:

[Ulises Sublime tutorial here]

More information on popular text editors can be found here:

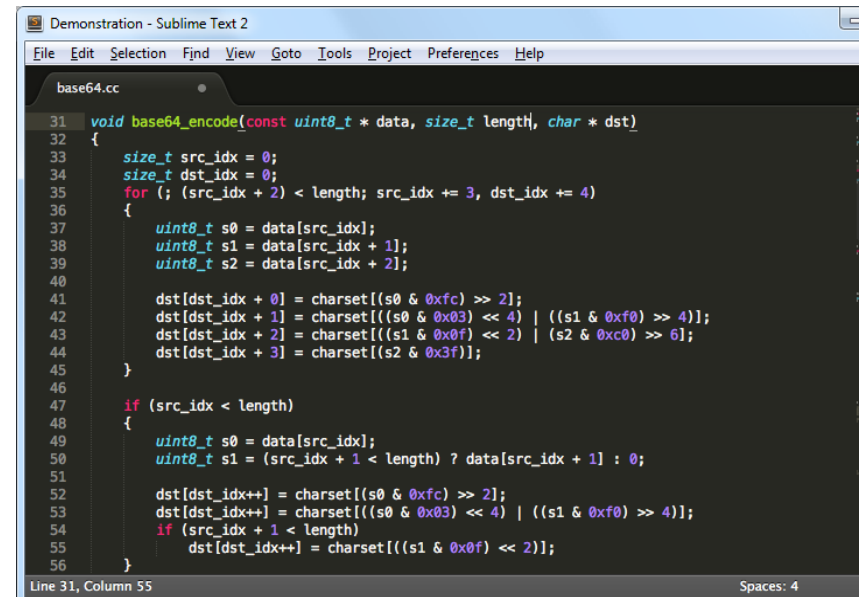
<http://lifehacker.com/five-best-text-editors-1564907215>



```

D:\source\notepad++\trunk\PowerEditor\src\EncodingMapper.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
langs.model.xml changelog EncodingMapper.cpp new 2
66 {932, "Shift_JIS MS_Kanji csShiftJIS csWindows31J"},
67 {949, "windows-949 korean"},
68 {51949, "euc-kr csEUCKR"},
69 {874, "tis-620"},
70 {10007, "x-mac-cyrillic xmaccyrillic"},
71 {21866, "koi8_u"},
72 {20866, "koi8_r csKOI8R"}
73 };
74
75 EncodingMapper * EncodingMapper::_pSelf = new EncodingMapper;
76
77 bool isInListA(const char *token, const char *list) {
78     if (!token || !list)
79         return false;
80     char word[64];
81     int i = 0;
82     int j = 0;
83     for (; i <= int(strlen(list)) ; i++)
84     {
85         if ((list[i] == ' ') || (list[i] == '\0'))

```



```

Demonstration - Sublime Text 2
File Edit Selection Find View Goto Tools Project Preferences Help
base64.cc
31 void base64_encode(const uint8_t * data, size_t length, char * dst)
32 {
33     size_t src_idx = 0;
34     size_t dst_idx = 0;
35     for (; (src_idx + 2) < length; src_idx += 3, dst_idx += 4)
36     {
37         uint8_t s0 = data[src_idx];
38         uint8_t s1 = data[src_idx + 1];
39         uint8_t s2 = data[src_idx + 2];
40
41         dst[dst_idx + 0] = charset[(s0 & 0xfc) >> 2];
42         dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
43         dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
44         dst[dst_idx + 3] = charset[(s2 & 0x3f)];
45     }
46
47     if (src_idx < length)
48     {
49         uint8_t s0 = data[src_idx];
50         uint8_t s1 = (src_idx + 1 < length) ? data[src_idx + 1] : 0;
51
52         dst[dst_idx++] = charset[(s0 & 0xfc) >> 2];
53         dst[dst_idx++] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
54         if (src_idx + 1 < length)
55             dst[dst_idx++] = charset[((s1 & 0x0f) << 2)];
56     }

```

Simple Polygon with Stroke

We will start off providing color for our base layer, DenverBoundary. We will make a white and black style for this layer to contrast with later products.

- 1) From the SLD Cookbook, under Polygons, click on *Simple polygon with stroke*.
- 2) Scroll down and click on *View and download the full "Simple polygon" SLD*
- 3) A screen with XML will appear. This does not contain the head of the code that will allow it to run. Right click and select Save as... Save it as an XML document with ".sld" as the extension.

- [Polygons](#)
 - [Example polygons layer](#)
 - [Simple polygon](#)
 - [Simple polygon with stroke](#)

Code

View and download the full "Simple polygon" SLD

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0.0"
  xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd">
  <NamedLayer>
    <Name>Simple polygon with stroke</Name>
    <UserStyle>
      <Title>SLD Cook Book: Simple polygon with stroke</Title>
      <FeatureTypeStyle>
        <Rule>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#000080</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#FFFFFF</CssParameter>
              <CssParameter name="stroke-width">2</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

File name: **polygon_simplepolygonwithstroke.sld**

Save as type: XML Document

4) Now, open the document in Sublime. It should look like this:

C:\Users\jrain_000\Downloads\polygon_simplepolygonwithstroke.sld.xml - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <StyledLayerDescriptor version="1.0.0"
3   xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
4   xmlns="http://www.opengis.net/sld"
5   xmlns:ogc="http://www.opengis.net/ogc"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
8   <NamedLayer>
9     <Name>Simple polygon with stroke</Name>
10    <UserStyle>
11      <Title>SLD Cook Book: Simple polygon with stroke</Title>
12      <FeatureTypeStyle>
13        <Rule>
14          <PolygonSymbolizer>
15            <Fill>
16              <CssParameter name="fill">#000080</CssParameter>
17            </Fill>
18            <Stroke>
19              <CssParameter name="stroke">#FFFFFF</CssParameter>
20              <CssParameter name="stroke-width">2</CssParameter>
21            </Stroke>
22          </PolygonSymbolizer>
23        </Rule>
24      </FeatureTypeStyle>
25    </UserStyle>
26  </NamedLayer>
27 </StyledLayerDescriptor>

```

This appears like HTML with hierarchical tags.

Quick lesson on tags

<FeatureTypeStyle>: Tells GeoServer this is styling a feature

<PolygonSymbolizer>: Tell GeoServer what type of feature

<Rule>: Instigates the rules for the style to follow

<Fill>: Determines the polygons fill color

<CssParameter>: Allows any CSS color code to be used for styling

<Stroke>: The border color

Reading and understanding tags will become more important as you create custom and more complex styles.

```
<FeatureTypeStyle>
  <Rule>
    <PolygonSymbolizer>
      <Fill>
        <CssParameter name="fill">#000080</CssParameter>
      </Fill>
      <Stroke>
        <CssParameter name="stroke">#FFFFFF</CssParameter>
        <CssParameter name="stroke-width">2</CssParameter>
      </Stroke>
    </PolygonSymbolizer>
  </Rule>
</FeatureTypeStyle>
```

- 5) Look up the hex code for white and black

https://www.w3schools.com/cssref/css_colors.asp

- 6) Set the fill to white and the stroke to black. Then, save the file as "polygon_whiteWBlackStroke"

```
<CssParameter name="fill">#ffffff</CssParameter>
</Fill>
<Stroke>
  <CssParameter name="stroke">#000000</CssParameter>
```

- 7) Next, change both the fill and stroke to black. Then, save as "polygon_blackWBlackStroke"

```
<CssParameter name="fill">#000000</CssParameter>
</Fill>
<Stroke>
  <CssParameter name="stroke">#000000</CssParameter>
```

- 8) Now, log back into GeoServer. Navigate to *Styles* and click *add new style*.

Styles

Manage the Styles published by GeoServer

-  Add a new style
-  Removed selected style(s)

- 9) Create the style, polygon_whiteWBlackStroke:
Name your style the same as the files name. Designate the style to be saved under your workspace, then scroll down and either upload your file, or copy and paste it into the *Style Editor*.

****Note:** if you upload your SLD, it will overwrite your style name with the file name.

- 10) Always validate! Once your code is in the *Style Editor*, click *Validate*. This let us know if the code can be understood by GeoServer. Scroll up to the top and look for validation errors. When you have no errors, click *Submit* to save the style.

- 11) Repeat for polygon_blackWBlackStroke

- 12) Now that the styles have been created, navigate to *Layers* and select *DenverBoundary*

- 13) Select the publishing tab

No validation errors.

New style

Upload a style file

Choose File polygon_sim...ke.sld.xml Upload ...

Style Editor

12pt ▼

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <StyledLayerDescriptor version="1.0.0"
3   xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescrip
4   xmlns="http://www.opengis.net/sld"
5   xmlns:ogc="http://www.opengis.net/ogc"
6   xmlns:xlink="http://www.w3.org/1999/xlink"
7   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
8   <NamedLayer>
9     <Name>Simple polygon with stroke</Name>
10    <UserStyle>
11      <Title>SLD Cook Book: Simple polygon with stroke</Title>
12      <FeatureTypeStyle>
13        <Rule>
14          <PolygonSymbolizer>
15            <Fill>
16              <CssParameter name="fill">#000080</CssParameter>
17            </Fill>
18            <Stroke>
19              <CssParameter name="stroke">#FFFFFF</CssParameter>
20              <CssParameter name="stroke-width">2</CssParameter>
21            </Stroke>
22          </PolygonSymbolizer>
23        </Rule>
24      </FeatureTypeStyle>
25    </UserStyle>
  
```

1 Validate 2 Apply Submit Cancel

- 14) Scroll down to WMS Settings and set the *Default Style* to `polygon_whiteWBlackStroke`. Then, add `polygon_whiteWBlackStroke` and `polygon_blackWBlackStroke` from the *Available Styles* to the *Selected Styles*. This will enable us to alternate between them as a WMS. Save.

WMS Settings

Layer Settings

☒ Queryable

☐ Opaque

Default Style

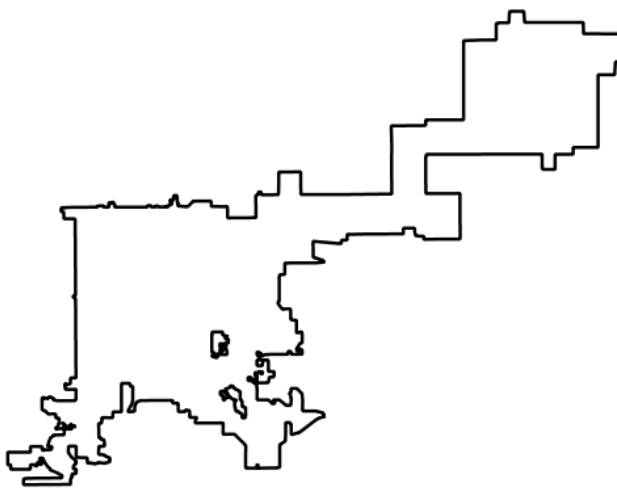
Raines_Workspace:polygon_whiteWBlackStroke



Additional Styles

Available Styles		Selected Styles
Raines_Workspace:point_simpler	⇌	Raines_Workspace:polygon_whiteWBlackStroke
Raines_Workspace:points_GraduatedSymbols	⇌	Raines_Workspace:polygon_blackWBlackStroke
Raines_Workspace:polygon_attributebasedpol		
Raines_Workspace:polygon_ByAttribute_Rain		
Raines_Workspace:polygon_polygonwithdefau		
Raines_Workspace:Polyline_StyledLabeled		
raster		
Raines_Workspace:raster_manycolorgradient2		
Raines_Workspace:raster_manycolorgradient_		
Red Line		
Raines_Workspace:ScaleCov		

- 15) Navigate to *Layer Preview* and view *DenverBoundary* with *OpenLayers*. Click the 3 horizontal dots in the top left corner to access this menu. From the *Styles* drop down menu, view your two styles.



Optimized and Styled Label

We will now be adding labeled roads to Denver.

- 1) Navigate to the SLD Cookbook and use the same process to download *optimized and styled label*
- 2) Open it your text editor. We will be changing the `ogc:PropertyName` (road name attribute) and the `CssParameter` for fill
- 3) In GeoServer, select *DenverArterials* from *Layers*. It should say “Edit Layer” at the top. Scroll to the bottom. You should see the attribute field names. Copy “NAME” and set it as your `ogc:PropertyName`.

```
<ogc:PropertyName>NAME</ogc:PropertyName>
```

- 4) Change the stroke (polyline color) to light grey and the fill (label color) to black

```
<CssParameter name="stroke">#d3d3d3</CssParameter>
```

```
<CssParameter name="fill">#000000</CssParameter>
```

- 5) Save it as “line_greyLineWLabel.sld

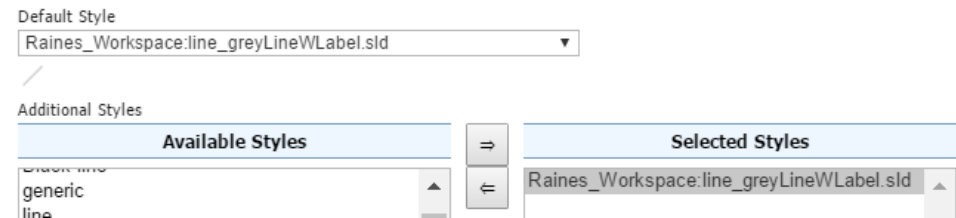
File name:	line_greyLineWLabel.sld
Save as type:	XML (*.xml;*.xsd;*.xslt;*.tld;*.dtml;*.rss;*.opml;*.svg)

- 6) In GeoServer, navigate to *Styles* and save it the same way as the previous style.

Feature Type Details

Property	Type	Nullable
the_geom	MultiLineString	true
TRAVLANES	Integer	true
LANEWIDTH	Integer	true
SPEEDLIMIT	Integer	true
Shape_Leng	Double	true
NAME	String	true

- 7) Similarly, add the style to the *DenverArterials* layer as the default and selected style



- 8) Now view your styled and labels roads on top of Denver by adding “DenverBoundary,” to your URL:

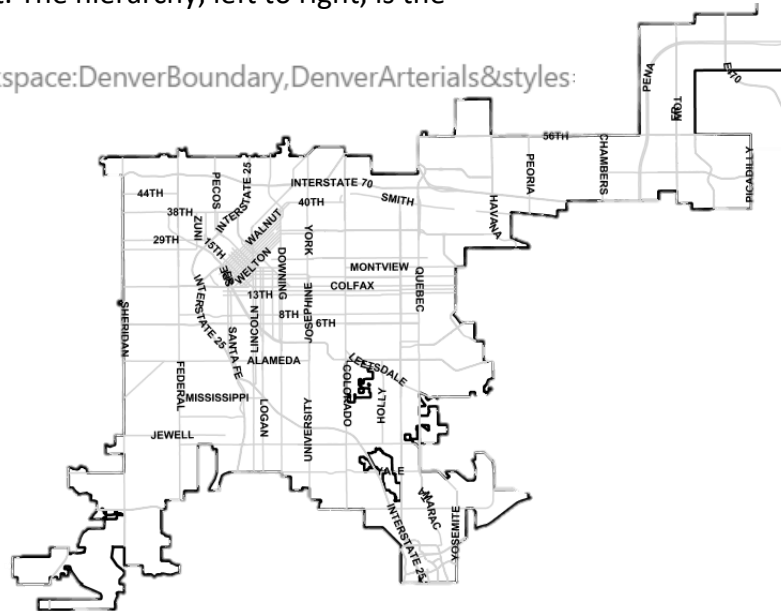
`GetMap&layers=Raines_Workspace:DenverArterials&styles`

This is where layers are in the URL. After “Workspace:” and before “&styles”.

`GetMap&layers=Raines_Workspace:DenverArterials&styles`

Add “DenverBoundary,” first. The hierarchy, left to right, is the drawing order.

`GetMap&layers=Raines_Workspace:DenverBoundary,DenverArterials&styles:`



Attribute-based point

Now we will be adding the *SpeedRecordings* to our current layers and styling them by attribute and with graduated symbols.

- 1) Navigate to the SLD Cookbook and use the same process to download *Attribute-based point*
- 2) Open it in your text editor. There are currently 3 rules for this style, we will add a fourth
- 3) Copy the entire Rule for MediumPop (from <Rule> to </Rule>) and paste it between the Rule for MediumPop and LargePop.
- 4) Change the <Name> for each Rule to the following:

SmallPop → UnderSpeedMajor

MediumPop → UnderSpeedMinor

MediumPop (the new one) → SpeedingMinor

LargePop → SpeedingMajor

- 5) Change all <ogc:PropertyName> to:

Speed_Diff

```

</Graphic>
</PointSymbolizer>
</Rule>
<Rule>
  <Name>MediumPop</Name>
  <Title>50000 to 100000</Title>
  <ogc:Filter>
    <ogc:And>
      <ogc:PropertyIsGreaterThanOrEqualTo>
        <ogc:PropertyName>pop</ogc:PropertyName>
        <ogc:Literal>50000</ogc:Literal>
      </ogc:PropertyIsGreaterThanOrEqualTo>
      <ogc:PropertyIsLessThan>
        <ogc:PropertyName>pop</ogc:PropertyName>
        <ogc:Literal>100000</ogc:Literal>
      </ogc:PropertyIsLessThan>
    </ogc:And>
  </ogc:Filter>
  <PointSymbolizer>
    <Graphic>
      <Mark>
        <WellKnownName>circle</WellKnownName>
        <Fill>
          <CssParameter name="fill">#0033CC</CssParameter>
        </Fill>
      </Mark>
      <Size>12</Size>
    </Graphic>
  </PointSymbolizer>
</Rule>
<Rule>
  <Name>LargePop</Name>
  <Title>Greater than 100000</Title>

```

- 6) Change <ogc:Literal> to the following for the respective Rule:
(Literal is the value in which arguments such as
<ogc:PropertyIsGreaterThanOrEqualTo> can be executed from)

UnderSpeedMajor

- <ogc:PropertyIsLessThan>:
 - <ogc:Literal>-6.83</ogc:Literal>

UnderSpeedMinor

- <ogc:PropertyIsGreaterThanOrEqualTo>:
 - <ogc:Literal>-6.83</ogc:Literal>
- <ogc:PropertyIsLessThan>:
 - <ogc:Literal>0</ogc:Literal>

SpeedingMinor

- <ogc:PropertyIsGreaterThanOrEqualTo>:
 - <ogc:Literal>0</ogc:Literal>
- <ogc:PropertyIsLessThan>:
 - <ogc:Literal>3</ogc:Literal>

SpeedingMajor

- <ogc:PropertyIsGreaterThanOrEqualTo>:
 - <ogc:Literal>3</ogc:Literal>

- 7) Then change the <Title> accordingly
- 8) Now we are going to change the size of the points and their color based on each category or Rule. Make the following changes:

UnderSpeedMajor

Fill: #0033CC

An example of UnderSpeedMinor after changing the
<ogc:PropertyName>,<ogc:Literal>, and <Title>

```
<Rule>
  <Name>UnderSpeedMinor</Name>
  <Title>-6.83 to 0</Title>
  <ogc:Filter>
    <ogc:And>
      <ogc:PropertyIsGreaterThanOrEqualTo>
        <ogc:PropertyName>Speed_Diff</ogc:PropertyName>
        <ogc:Literal>-6.83</ogc:Literal>
      </ogc:PropertyIsGreaterThanOrEqualTo>
      <ogc:PropertyIsLessThan>
        <ogc:PropertyName>Speed_Diff</ogc:PropertyName>
        <ogc:Literal>0</ogc:Literal>
      </ogc:PropertyIsLessThan>
    </ogc:And>
  </ogc:Filter>
  <PointSymbolizer>
    <Graphic>
      <Mark>
        <WellKnownName>circle</WellKnownName>
        <Fill>
          <CssParameter name="fill">#0033CC</CssParameter>
        </Fill>
      </Mark>
      <Size>12</Size>
    </Graphic>
  </PointSymbolizer>
</Rule>
```


Size: 6
 UnderSpeedMinor
 Fill: #0033CC
 Size: 8
 SpeedingMinor
 Fill: #ff0000
 Size: 12
 SpeedingMajor
 Fill: #ff0000
 Size: 16

Now we will see speed increase with the size of the point and where people drive under the speed limit will be blue and over the speed limit, red.

- 9) Save your SLD as point_speedRecordingsBlueRed.sld
- 10) Open GeoServer and turn this into a new style, then add it to the *SpeedRecordings* layer under the publisher tab as the default style. It should appear something similar to this under the WMS Settings. It will show the <Title>s you've given the Rules.
- 11) In GeoServer, navigate to *Layer Groups*.
- 12) Create a new layer group
 - a. Use the *Coordinate Reference System* EPSG:2773
 - b. Generate Bounds
 - c. Add DenverBoundary, DenverArterials, and SpeedRecordings in this drawing order. The higher the

- -16.84 to -6.83
- -6.83 to 0
- 0 to 3
- Greater than 3

Layers

+ Add Layer...

+ Add Layer Group...

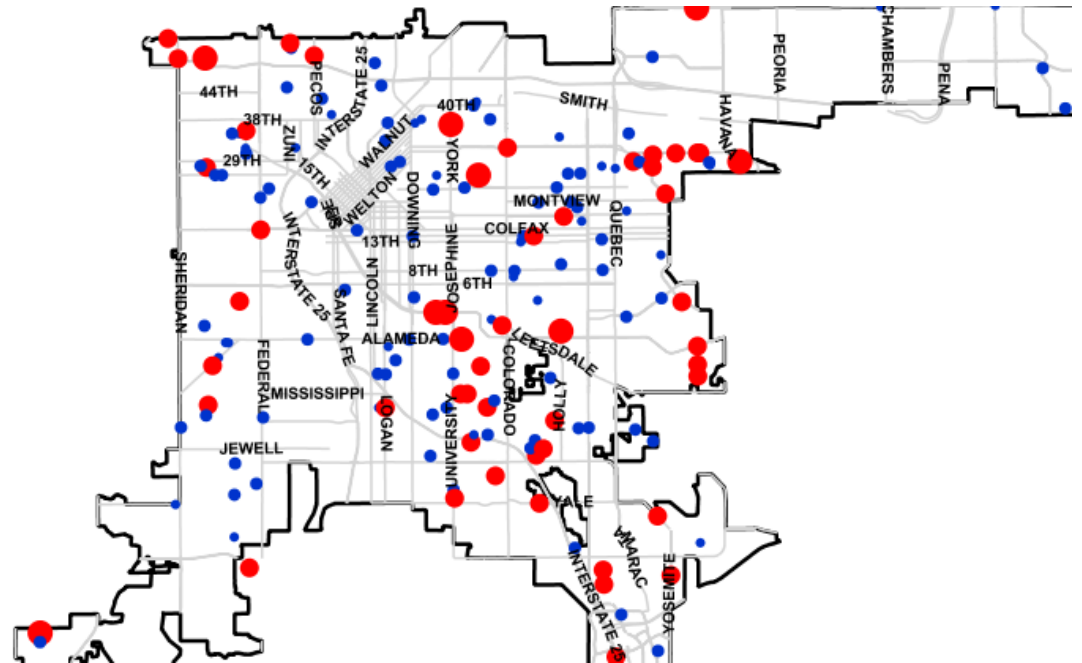
Drawing order	Layer	Default Style
1 ↓	Raines_Workspace:DenverBoundary	<input type="checkbox"/>
2 ↑ ↓	Raines_Workspace:DenverArterials	<input type="checkbox"/>
3 ↑	Raines_Workspace:SpeedRecordings	<input type="checkbox"/>

<< < 1 > >> Results 0 to 0 (out of 0 items)

drawing order number, the more forward, or on top, the layer will be.

d. Save.

13) View the layer group in *Layer Preview*



Many Color Gradient

Now we will style a raster to view the density of vehicle crashes in Denver.

- 1) Change the default style of *DenverBoundary* to *polygon_blackWBlackStroke*
- 2) Navigate to the SLD Cookbook and use the same process to download *Many color gradient*

- 3) This raster represents an index with very small values. The values in this SLD are represented by “quantity”

```
<ColorMapEntry color="#000000" quantity="95" />
```

Change all of the quantities as follows:

```
quantity=".0000005008"
quantity=".000001878"
quantity=".000038813"
quantity=".000063853"
quantity=".000096405"
quantity=".000138974"
quantity=".000194063"
quantity=".000312"
```

- 4) The first *ColorMapEntry* will go beyond the Denver boundaries, so set it to 0 opacity:

```
<ColorMapEntry color="#000000" quantity=".0000005008" opacity='0' />
```

Your SLD should now look like this:

```
<ColorMap>
  <ColorMapEntry color="#000000" quantity=".0000005008" opacity='0' />
  <ColorMapEntry color="#0000FF" quantity=".000001878" />
  <ColorMapEntry color="#00FF00" quantity=".000038813" />
  <ColorMapEntry color="#FF0000" quantity=".000063853" />
  <ColorMapEntry color="#FF00FF" quantity=".000096405" />
  <ColorMapEntry color="#FFFF00" quantity=".000138974" />
  <ColorMapEntry color="#00FFFF" quantity=".000194063" />
  <ColorMapEntry color="#FFFFFF" quantity=".000312" />
</ColorMap>
```

- 5) Save it and add it as the default style to your layer, *VehCrashes_HeatMap*

- 6) Create a Layer Group with DenverBoundary and VehCrashes_HeatMap
- 7) View your new Layer Group in Layer Preview

