GeoServer
Cartographic Rendering

New features for map makers

Andrea Aime
aaime@opengeo.org
Three SLDs
SLD 1.0, 1.1, and GS one

SE 1.1 and SLD 1.1

SLD 1.0

SLD 1.0 + GS extensions
**SE 1.1 improvements**

- Symbolizers in real world units (uom)
- Selected geometry transformations: offsets, buffers
- External symbol sets (support for decoravite fonts)
- Functions: numeric, date and string formatting, categorization, interpolation, and recoding
**GeoServer improvements**

- GeoServer extended SLD 1.0 over time by adding a number of vendor extensions
- Some shared with SE 1.1, some unique
- That's the content of this presentation!
Summary

- Recent improvements
- Filter functions
- Geometry transformations
- Labeling

1.0  SLD 1.0
1.1  SLD 1.1 only
GS  GeoServer specific

2.0  GeoServer 2.0.x
2.1  GeoServer 2.1.x (trunk)
Improvements

Things we were missing or not doing quite right
Graphic strokes (finally!)

- Graphic stroke: replicate an image along a line

```xml
...<LineSymbolizer>
  <Stroke>
    <GraphicStroke>
      <Graphic>
        <ExternalGraphic>
          <OnlineResource xlink:type="simple" xlink:href="burg02.svg"/>
          <Format>image/svg+xml</Format>
        </ExternalGraphic>
        <Size>
          <ogc:Literal>20</ogc:Literal>
        </Size>
      </Graphic>
    </GraphicStroke>
  </Stroke>
</LineSymbolizer>
..."
2.0.3

2.1.x (trunk)

Improved quality

No cut symbols
• “The stroke-dasharray element encodes a dash pattern as a series of space separated floats”
• What about mixing dash array with graphic stroke? Spec does not say...
<FeatureTypeStyle>
  <Rule>
    <LineSymbolizer>
      <Stroke>
        <GraphicStroke>
          <Graphic>
            <WellKnownName>Circle</WellKnownName>
            <Mark>
              <Fill><CssParameter name="fill">#FFFFFF</CssParameter></Fill>
              <Size>5</Size></Mark></Graphic>
          </GraphicStroke>
          <CssParameter name="stroke-dasharray">5 35</CssParameter>
        </Stroke>
      </LineSymbolizer>
    </Rule>
  </FeatureTypeStyle>
  <Rule>
    <LineSymbolizer>
      <Stroke>
        <GraphicStroke>
          <Graphic>
            <WellKnownName>Star</WellKnownName>
            <Mark>
              <Fill><CssParameter name="fill">#FFFFFF</CssParameter></Fill>
              <Size>10</Size></Mark></Graphic>
          </GraphicStroke>
          <CssParameter name="stroke-dasharray">10 30</CssParameter>
        </Stroke>
      </LineSymbolizer>
    </Rule>
  </FeatureTypeStyle>
</FeatureTypeStyle>
Full vector export

- SVG and PDF formats now provide full vector output
- Requires that all graphics are vector themselves: marks or SVG symbols
- History:
  - In 1.7.x all graphics were rasterized
  - In 2.0.x support vector output of point and polygon fills thanks to Milton Jonathan work
  - In trunk 2.1.x complete support (graphic strokes as well)
Unit Of Measure

- SLD 1.0 supports only pixels
- SLD 1.1 has a `uom` attribute: pixels, meters or feet
- GeoServer SLD 1.0 accepts the UOM attribute anyways (thanks again to the work of Milton Jonathan)

```xml
<LineSymbolizer uom="http://www.opengeospatial.org/se/units/metre">
  <Stroke>
    <CssParameter name="stroke-width">500</CssParameter>
  </Stroke>
</LineSymbolizer>
```
Increasing zoom level
The twilight zone

Stuff that is part of the SLD specification, yet it's not portable
Leveraging SLD flexibility

- In SLD most elements are of the type `ogc:Expression`
  - Attribute names
  - Math (`ogc:Add`, `ogc:Div`, ...)
  - Call functions!
- Functions are open ended!

\[ e = f(x, y, z) \]
Filter functions

• The **concept** of filter function is **part of the OGC Filter spec**. A filter function is an expression with a name and a set of arguments.

• However there are **no standardized functions in SLD 1.0**, and only a handful in SE 1.1.

Filter function examples

- **Math**: abs, sin, cos, tan, floor, round, random, toDegrees, toRadians, ...
- **String**: strEqualsIgnoreCase, strLength, strReplace, strSubstring, strToLowerCase, strToUppercase, ...
- **Parsing and formatting**: dateFormat, numberFormat, ...
- **Geometry ones**: intersects, union, ...
Float point field, would result in 620.0, sometimes in 619.999999999
<Label>
  <ogc:Function name="strToUppercase"/>
  <ogc:PropertyName>STATE_NAME</ogc:PropertyName>
</ogc:Function>
</Label>
Format("#.##", UNEMPLOY / (EMPLOYED / UNEMPLOY))
Geometry transformations

Not your grandpa's geometries
Geometry reference in SLD

- Each SLD/SE symbolizer has a “Geometry” element
- Used if you have many geometries among the attributes (not common)
- Has to be a `<ogc:PropertyName>`
- Why? Can't I play with my geometry?
Geometry transformations

• In GeoServer extended SLD, `<Geometry>` can be ogc:Function too
• You can transform the geometry before the renderer starts using it
• Extract vertexes, centroid, buffer, translate, intersect, ...
<LineSymbolizer>
  <Stroke>
    <CssParameter name="stroke-width">0.5</CssParameter>
  </Stroke>
</LineSymbolizer>
<PointSymbolizer>
  <Geometry>
    <ogc:Function name="vertices">
      <ogc:PropertyName>the_geom</ogc:PropertyName>
    </ogc:Function>
  </Geometry>
  <Graphic>
    <Mark>
      <WellKnownName>circle</WellKnownName>
      <Fill>
        <CssParameter name="fill">#FF0000</CssParameter>
      </Fill>
      <Size>6</Size>
    </Mark>
  </Graphic>
</PointSymbolizer>
<PolygonSymbolizer>
  <Geometry>
    <ogc:Function name="offset">
      <ogc:PropertyName>the_geom</ogc:PropertyName>
      <ogc:Literal>0.00004</ogc:Literal>
      <ogc:Literal>-0.00004</ogc:Literal>
    </ogc:Function>
  </Geometry>
  <Fill><CssParameter name="fill">#555555</CssParameter></Fill>
</PolygonSymbolizer>

<PolygonSymbolizer>
  <Fill><CssParameter name="fill">#ff8878</CssParameter></Fill>
</PolygonSymbolizer>
Place a closed arrow at the end of the line

Rotate it along the line
The shape mark factory

- Simple general use shapes:
  - shape://vertline
  - shape://horline
  - shape://slash
  - shape://backslash
  - shape://dot
  - shape://plus
  - shape://times
  - shape://oarrow
  - shape://carrow
<LineSymbolizer><Stroke/></LineSymbolizer>

<LineSymbolizer>
  <Stroke>
    <GraphicStroke>
      <Graphic>
        <Mark>
          <WellKnownName>shape://vertline</WellKnownName>
          <Stroke/>
        </Mark>
        <Size>7</Size>
      </Graphic>
    </GraphicStroke>
  </Stroke>
</LineSymbolizer>

<PolygonSymbolizer>
  <Fill>
    <GraphicFill>
      <Graphic>
        <Mark>
          <WellKnownName>shape://slash</WellKnownName>
          <Stroke />
        </Mark>
        <Size>10</Size>
      </Graphic>
    </GraphicFill>
  </Fill>
  <Stroke/>
</PolygonSymbolizer>

<PolygonSymbolizer>
  <Fill>
    <GraphicFill>
      <Graphic>
        <Mark>
          <WellKnownName>shape://slash</WellKnownName>
          <Stroke />
        </Mark>
        <Size>20</Size>
      </Graphic>
    </GraphicFill>
  </Fill>
  <Stroke/>
</PolygonSymbolizer>
Map labeling

1001 vendor options
**SLD/SE status**

- SLD/SE provides control for label along a line and position relative to a point
- Quite poor. What about:
  - Priority
  - Repetition
  - Label wrapping
  - Controlling placement heuristics
  - Mixing labels and graphics so that they behave as one (road plates)
GeoServer status

• More than a dozen vendor options to control and fine tune labeling

• Full list here: http://docs.geoserver.org/trunk/en/user/styling/sld-reference/labeling.html
Controlling priority

- `<Priority>` vendor element
- The higher the value, the sooner the label will be drawn (which makes it win in the conflict resolution game)

```
<Priority><ogc:PropertyName>POP2005</ogc:PropertyName></Priority>
```
Controlling label wrapping

- An option to wrap labels that exceed a certain length, in pixels

```xml
<VendorOption name="autoWrap">100</VendorOption>
```
Repeating and displacing

- Over long lines it's better to repeat the labels
- Displacing makes GS look for other places should the candidate label position be busy

```xml
<VendorOption name="followLine">true</VendorOption>
<VendorOption name="maxDisplacement">50</VendorOption>
<VendorOption name="repeat">300</VendorOption>
```
Showing one way

• Labels are usually flipped to make them readable.
If the char happens to be a directional arrow... that's not desirable

```xml
<TextSymbolizer>
  <Label>&#x2129;</Label>
  <Font>
    <CssParameter name="font-family">OpenSymbol</CssParameter>
    <CssParameter name="font-size">10</CssParameter>
    <CssParameter name="font-weight">bold</CssParameter>
  </Font>
  <LabelPlacement>
    <LinePlacement />
  </LabelPlacement>
  <Halo>
    <Radius>
      <ogc:Literal>1</ogc:Literal>
    </Radius>
    <Fill>
      <CssParameter name="fill">#FFFFFF</CssParameter>
      <CssParameter name="fill-opacity">0.85</CssParameter>
    </Fill>
  </Halo>
  <Fill>
    <CssParameter name="fill">#AAAAAA</CssParameter>
  </Fill>
  <VendorOption name="maxDisplacement">100</VendorOption>
  <VendorOption name="forceLeftToRight">false</VendorOption>
</TextSymbolizer>
```
Mixing labels with graphics

- Typical case: road plate
- Either the road plate and the label show together, or none of them should
- Solution: include a Graphic element inside the TextSymbolizer!
Problem: the graphic size is fixed, the text one is dynamic! We could stretch it...
Resize mode: none, proportional, stretch
Questions?
Extras
Lightning intro to SLD
**SLD basic elements**

- **Style**: describes how a layer is to be depicted
  - **FeatureTypeStyle**
    - **Rule**
      - **Symbolizer**
**FeatureTypeStyle**

- “The FeatureTypeStyle defines the styling that is to be applied to a single feature type”
- “A map styler is expected to process all FeatureTypeStyles in the order that they appear, regardless, plotting one instance over top of another” (painter model)
- → Used mostly to force certain drawing order
Rule

• “Rules are used to group rendering instructions by feature-property conditions and map scales”

• So:
  – Scale dependencies
  – Filter by attribute
  – Rendering instructions that apply under the above conditions → symbolizers
Symbolizer

• “A Symbolizer describes how a feature is to appear on a map. The Symbolizer describes not just the shape that should appear but also such graphical properties as color and opacity.”

• Five types of symbolizers:
  – Point: symbol, size, color, ...
  – Line: width, color, graphics along a line
  – Polygon: outline, fill (solid color or graphic based)
  – Text: label, font, placement
  – Raster: color table, gamma, histogram, ...

• A rule can contain multiple symbolizers
Dynamic symbolizers

Breaking out the mark and graphic cage
Marks in SLD/SE

- Mark: a shape to be filled and stroked
- SLD 1.0:
  - “square”, “circle”, “triangle”, “star”, “cross”, and “x”
- SE 1.1: same, but also external symbol source and “mark index” (e.g. a decorative font + index inside of it)
Marks in GeoServer

• The well known name is a string, so it's open ended
• Our convention: factory://name
• Two factories available today:
  – shape
  – ttf
• More could be implemented, the API is pluggable
The shape mark factory

• Shapes intended to be hatch generators:
  – shape://vertline
  – shape://horline
  – shape://slash
  – shape://backslash
  – shape://dot
  – shape://plus
  – shape://times
  – shape://oarrow
  – shape://carrow
<LineSymbolizer><Stroke/></LineSymbolizer>

<PolygonSymbolizer>
  <Fill>
    <GraphicFill>
      <Mark>
        <WellKnownName>shape://vertline</WellKnownName>
        <Stroke/>
      </Mark>
      <Size>7</Size>
    </GraphicFill>
  </Fill>
</PolygonSymbolizer>

<LineSymbolizer>
  <Stroke>
    <GraphicStroke>
      <Graphic>
        <Mark>
          <WellKnownName>shape://slash</WellKnownName>
          <Stroke/>
        </Mark>
        <Size>10</Size>
      </Graphic>
    </GraphicStroke>
  </Stroke>
</LineSymbolizer>

<PolygonSymbolizer>
  <Fill>
    <GraphicFill>
      <Mark>
        <WellKnownName>shape://slash</WellKnownName>
        <Stroke/>
      </Mark>
      <Size>20</Size>
    </GraphicFill>
  </Fill>
</PolygonSymbolizer>
The TTF mark factory

• Generates shapes out of decorative fonts
• Format is ttf://fontname#charcode

```xml
<PointSymbolizer>
  <Graphic>
    <Mark>
      <WellKnownName>ttf://Webdings#0x0051</WellKnownName>
      <Fill>
        <CssParameter name="fill">#000000</CssParameter>
      </Fill>
    </Mark>
    <Size>20</Size>
  </Graphic>
</PointSymbolizer>
```
External graphics

• URL to an image
• URL cannot have parameters → static image only!
• Compare with Google chart API → dynamic image!

http://chart.apis.google.com/chart?
cht=p3&chd=s:Uf9a&chs=250x100
&chl=January|February|March|April
Enter dynamic symbolizers

• Dynamic symbolizers: expand attribute names inside mark names and graphic URLs

• Expand full CQL expressions (making math, formatting strings, calling functions)

• `${expression}/ ${attributeName}`
Calling a filter function to lower case the state abbreviation

```xml
<ExternalGraphic>
  <OnlineResource xlink:type="simple"
    xlink:href="http://www.usautoparts.net/tn_${strToLowerCase(STATE_ABBR)}.jpg" />
  <Format>image/jpeg</Format>
</ExternalGraphic>
```