Java Topology Suite in Action
Combining ESRI and Open Source

Jared Erickson
Pierce County, WA
Introduction

- Combing ESRI and Open Source GIS
- The Java Topology Suite (JTS)
- How Pierce County uses it with ESRI software
- How Pierce County extends it
ESRI and Open Source
What is JTS?

- Java API for Vector Geometry
  - Geometry Object Model
  - Geometry functions, Spatial Predicates, Overlay Methods, and Algorithms
- Open source
- Written by Martin Davis of Refractions Research
- Implements OGC Simple Features for SQL specification (no curves)
What is it?

- Core library in Java tribe of open source GIS
- JTS code is ported to C/C++ as GEOS
  - GEOS is the core library of the C open source tribe
  - GEOS is used by PostGIS, GDAL/OGR, MapServer, QGIS, Shapely (Python)
- Ported to .NET as NetTopologySuite
- Explicit Precision Model
- Focuses on Robustness vs. Speed
Geometry

Point

LineString

Polygon

MultiPoint

MultiLineString

MultiPolygon

GeometryCollection
Geometry Code Sample

GeometryFactory geometryFactory = new GeometryFactory();

Point point = geometryFactory.createPoint(new Coordinate(200.0, 323.0));

LineString lineString = geometryFactory.createLineString(new Coordinate[] {
    new Coordinate(2.2,3.3),
    new Coordinate(4.4,5.5),
    new Coordinate(6.6,7.7)
});
Spatial Functions and Predicates

- Buffer
- Contains
- ConvexHull
- CoveredBy
- Covers
- Crosses
- Difference
- Disjoint
- Distance
- Equals
- Area
- Boundary
- Centroid
- Envelope
- EnvelopeInternal
- Length
- Intersection
- Intersects
- Is Empty
- Is Simple
- Is Valid
- Is Within Distance
- Normalize
- Overlaps
- Relate (DE-9IM Intersection Matrix)
- SymDifference
- Touches
- Union
- Within
Spatial Functions and Predicates

GeometryFactory geometryFactory = new GeometryFactory();

Point point = geometryFactory.createPoint(new Coordinate(200.0, 323.0));
Geometry bufferedPoint = point.buffer(1000.00);

LineString lineString = geometryFactory.createLineString(new Coordinate[] {
    new Coordinate(2.2, 3.3),
    new Coordinate(4.4, 5.5),
    new Coordinate(6.6, 7.7)
});
Geometry envelope = lineString.getEnvelope();
Algorithms

- Validation
- Line Merging
- Polygonization
- Spatial Indexes (Quad Tree, STRtree, BinTree...)
- Linear Referencing
- Planar graphs
- Simplification (Douglas Peucker, Topology Preserving)
Who uses it?

GeoTools
The open source Java GIS toolkit

GeoServer

uDig
User-friendly Desktop Internet GIS

OPENJump
Java Unified Mapping Platform

Latitude Geographics™

Pierce County
WASHINGTON
JTS usage in Pierce County

- CountyView Web
- Pierce County GIS Web Services
- JTSIO
- JTS Web Processing
CountyView Web

- Entry Level Enterprise GIS
- Built on IMF, ArcIMS, ArcSDE
- Data Menu, Locator, Owner Notify, Metadata, Census, Printing, Open @

Focus:
- Data Viewer
- Printing
- Integration
- Ease of use
CountyView Web

- All geometry in AXL requests is translated to JTS geometry by the IMF framework

- Owner Notify

- Reverse Geocode

- Profile
CountyView Web: Owner Notify
Buffer the user’s selected map feature:

Geometry bufferedGeometry = geometry.buffer(bufferDistance);

Use the buffered geometry to perform a spatial query to return parcels

If using lots deep option, union all parcel geometry and buffer by 1 to get adjacent parcels

Geometry[] geom = new Geometry[geometries.size()];
geometries.toArray(geom);
GeometryFactory fact = geom[0].getFactory();
Geometry geomColl = fact.createGeometryCollection(geom);
Geometry union = geomColl.buffer(0);
Geometry bufferedGeometry = union.buffer(1);

Perform another spatial query using the buffered geometry to get parcels
CountyView Web: Reverse Geocode
CountyView Web: Reverse Geocode

- Perform spatial query around an x,y at some distance on roads layer.
- Get the closest Geometry (LineString)

Coordinate coordinate = new Coordinate(x,y);

// Put the user’s Coordinate on the LineString
LocationIndexedLine lineRef = new LocationIndexedLine(lineString);
LinearLocation loc = lineRef.project(coordinate);
Coordinate coordOnLine = loc.getCoordinate(lineString);

// Figure out how far the Coordinate is along the LineString
LengthLocationMap locationMap = new LengthLocationMap(lineString);
double distanceAlong = locationMap.getLength(loc);
double lineLength = lineString.getLength();
double percentAlong = distanceAlong / lineLength;

// Use percentAlong to interpolate between to and from address ranges
CountyView Web: Profiles
Perform spatial query getting all contours that intersect with a LineString draw by the user.

// Index the profile LineString for linear referencing
LocationIndexedLine lineRef = new LocationIndexedLine(lineString);
double lineLength = lineString.getLength();

// For each contour, find the intersection between the users LineString and
// the contour
Geometry intersection = lineString.intersection(geometry);

// Get the coordinate of this intersection and calculate the distance along and percent
// long the LineString
Coordinate coordinate = ((Point)intersection.getGeometryN(i)).getCoordinate();
LinearLocation loc = lineRef.project(coordinate);
LengthLocationMap locationMap = new LengthLocationLocationMap(lineString);
double distanceAlong = locationMap.getLength(loc);
double percentAlong = distanceAlong / lineLength;
GIS Web Services

Geocoders

1. Address Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it] [Try it (intersection)] [Try it (long form)]
2. Address With City and Zip Code Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
3. Address Geocoder 2 (experimental) [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
4. Basemap Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
5. Business by Division Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
6. Business by Name Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
7. Business by SIC Code Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
8. 2000 Census Tract Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
9. 2000 Census Block Group Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
10. 2000 Census Block Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
11. Child Care Center Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
12. Community Plan Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
13. County Facility Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
14. Drainage Subsystem Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
15. Hospital Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
16. Intersection Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
17. Mobility Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
18. Parcel Number Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
19. Parcel Tax Payer Name Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
20. Park Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
21. Patrol District Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
22. Patrol Reporting Block Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
23. PC-NET/NAT Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
24. Place Name Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
25. Plat Number Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
26. Plat Name Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
27. RTSO Geocoder [WSDL] [XML] [CSV] [KML] [JSON] [Try it]
GIS Web Services

- Pierce County GIS web services
  - Geocoders, Spatial Queries, Projection, Open @, Tile Caches, Data
  - SOAP, XML, CSV, JSON, KML

- Documentation and Examples
  - Suggested user interfaces for non GIS developers

- Serves thousands of requests to 12 applications across 3 departments.
GIS Web Services

- Common Geometry model between ArcSDE and ArcIMS for Spatial Queries

- Reprojection
  - JTS and GeoTools
GIS Web Services: Spatial Queries

- Buffer service
  - Parameters Point, Distance, Layer
  - Buffer the Point by the Distance
  - Turn buffered Polygon into ArcIMS AXL or ArcSDE Geometry (or WKT for PostGIS)
  - Perform spatial query
  - Turn AXL, ArcSDE Geometry, WKT back into JTS Geometry
GIS Web Services: Projection

- Uses JTS and GeoTools
- Uses European Petroleum Survey Group (EPSG)
  - EPSG:4326 is WGS 84
  - EPSG:2927 is WA State Plane South (feet)

```java
CoordinateReferenceSystem fromCRS = CRS.decode(fromEPSG);
CoordinateReferenceSystem toCRS = CRS.decode(toEPSG);

MathTransform math = CRS.findMathTransform(fromCRS, toCRS);
DirectPosition pos = new GeneralDirectPosition(x, y);
DirectPosition geoPos = math.transform(pos, null);
```
JTSIO

- Java Library for reading and writing to and from JTS Geometry Objects and geometry string formats

- JTS to AXL
  - Point(10,20) to <POINT x="10.0" y="20.0" />

- Other Neogeoography/Web 2.0 geometry string formats
  - KML
  - GeoJSON
  - GeoRSS
  - GML
  - GPX
  - WKT
Format Examples

**WKT**

```
POINT(1 1)
```

**AXL**

```
<POINT x="1.0" y="1.0" />
```

**GeoJSON**

```
{"type":"Point","coordinates":[1,1]}
```

**GeoRSS**

```
<georss:point>1.0 1.0</georss:point>
```

**GML**

```
<gml:Point xmlns:gml="http://www.opengis.net/gml">
  <gml:coordinates>1.0,1.0</gml:coordinates>
</gml:Point>
```

**GPX**

```
<wpt xmlns="http://www.topografix.com/GPX/1/1" lat="1.0" lon="1.0"/>
```

**KML**

```
<Point><coordinates>1.0,1.0</coordinates></Point>
```
## JTSIO Geometry/Format Matrix

<table>
<thead>
<tr>
<th></th>
<th>axl</th>
<th>geo Json</th>
<th>geo JsonDoc</th>
<th>Geo Rss</th>
<th>Geo Rss Doc</th>
<th>Geo Rss Gml</th>
<th>gml</th>
<th>gmlDoc</th>
<th>gpx</th>
<th>Gpx Doc</th>
<th>kml</th>
<th>Kml Doc</th>
<th>wkt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Point</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Multi Point</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Line String</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Linear Ring</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Polygon</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Multi LineString</strong></td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Multi Polygon</strong></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Geometry Collection</strong></td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
JTSIO in action

GeometryFactory geometryFactory = new GeometryFactory();
Coordinate coord = new Coordinate(5.1, 5.2);
Point point = geometryFactory.createPoint(coord);

GeoJsonJtsWriter writer = new GeoJsonJtsWriter();
String geoJson = writer.writePointToString(point);

{"type":"Point","coordinates":[5.1,5.2]}

GeoJsonJtsReader reader = new GeoJsonJtsReader();
Point point2 = reader.readPoint(geoJson);
JTS Web Processing
JTS Web Processing

- RESTful Geoprocessing Server that exposes JTS spatial operators
  - Buffer, intersection, union, touches, etc...
- Geometry had to be encoded in text based format using JTSIO
- Idea based on Christopher Schmidt’s WebProcessingServer
- 40 or so web services
- Consumable from Javascript via Ajax
- Displayable in OpenLayers
JTS Web Processing: How it works

Request
http://whitneydev.co.pierce.wa.us/jts/services/wkt/buffer?
geometry=POINT (108420.33
753808.59)&distance=500&quadrantSegments=8&capStyle=round&sourceCRS=EPSG:2927&targetCRS=EPSG:2927

Response
POLYGON ((108920.33 753808.59, 108910.72264020162 753711.044838992,
108882.26976625565 753617.2482838174, 108836.06480615128 753530.8048834902,
108773.88339059327 753455.0366094067, 108698.1151165098 753392.8551938487,
108611.67171618255 753346.6502337443, 108517.87516100807 753318.1973597984,
108420.33 753308.59, 108322.78483899194 753318.1973597984, 108228.98828381745
753346.6502337443, 108142.5448834902 753392.8551938487, 108066.77660940673
753455.0366094067, 108004.59519384873 753530.8048834902, 107958.39023374436
753617.2482838174, 107929.93735979838 753711.044838992, 107920.33 753808.59,
107929.93735979838 753906.135161008, 107958.39023374436 753999.9317161825,
108004.59519384873 754086.3751165097, 108066.77660940673 754162.1433905932,
108142.5448834902 754224.3248061512, 108228.98828381745 754270.5297662556
108322.78483899194 754298.9826402016, 108420.33 754308.59, 108517.87516100807
754298.9826402016, 108611.67171618255 754270.5297662556, 108698.1151165098
754224.3248061512, 108773.88339059327 754162.1433905932, 108836.06480615128
754086.3751165097, 108882.26976625565 753999.9317161825, 108910.72264020162
753906.135161008, 108920.33 753808.59))
/**
 * Create a custom buffer point tool
 */
function createBufferPointTool() {
    var tool = new OpenLayers.Control();
    OpenLayers.Util.extend(tool, {
        draw: function () {
            this.handler = new OpenLayers.Handler.Point(tool, {"done": this.notice});
        },
        notice: function (pt) {
            var distanceStr = prompt("Enter buffer distance:");
            if (!distanceStr || isNaN(distanceStr)) {
                alert("Please enter a distance!");
                return;
            }
            var distance = parseFloat(distanceStr);
            var geoJsonFormat = new OpenLayers.Format.GeoJSON();
            var geoJson = geoJsonFormat.write(pt, false);
            $.ajax({
                type: "POST",
                url: "/jts/services/geoJson/buffer",
                data: {
                    "geometry": geoJson,
                    "distance": distance
                },
                dataType: 'json',
                success: function(json){
                    var geoJsonFormat = new OpenLayers.Format.GeoJSON();
                    var feature = geoJsonFormat.read(json);
                    if (json) {
                        vlayer.addFeatures(feature);
                    }
                },
                error: function(request, text, error) {
                    alert("Error buffering point!");
                }
            });
        },
        CLASS_NAME: "OpenLayers.Control.BufferPointTool",
        displayClass: "olControlBufferPointTool",
        type: OpenLayers.Control.TYPE_TOOL
    });
    return tool;
}
JTS Web Processing and OpenLayers
Conclusion

- GIS Software Development doesn’t have to be either (ESRI) or (Open Source)

- JTS is an incredibly powerful and useful library

- JTS can easily extend ArcIMS and ArcSDE