Make your own map-based mashup
Create a KML service from ESRI shapefile data

Shyam Sundar Nagarajan

Map-based mashups abound these days. Mashups require services that can be mashed up. Location-based mashups need services that provide boundary information. With Web-based mapping providers, you can easily create a map-based mashup with little or no capital investment. In this article, learn how to create a KML boundary service from an ESRI shapefile to be used in mashups.

Introduction

Map-based mashups are omnipresent nowadays, largely because services that can be mashed up are widely available. For a map-based mashup, you need services that provide cartographic boundaries. The U.S. Census Bureau releases data on state, county, and ZIP code boundaries in Environmental Systems Research Institute (ESRI) shapefile format. In this article, learn how to create a simple service to deliver KML files from ESRI shapefiles using open source software.

Technology and software products

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIS</td>
<td>Geographical information systems</td>
</tr>
<tr>
<td>KML</td>
<td>Keyhole markup language</td>
</tr>
<tr>
<td>Mashup</td>
<td>A Web hybrid application that combines data from more than one source into an integrated experience</td>
</tr>
<tr>
<td>TIGER</td>
<td>Topologically integrated geographic encoding and referencing</td>
</tr>
</tbody>
</table>

There are many tools and products, both commercial and open source, that you can use to work with spatial data and create mashups. The example in this article uses open source software: PostGIS as the database, GeoTools Java™ library, the OpenLayers mapping library, and Apache Tomcat as the application server.

PostGIS

A spatial extension to the open source relational database PostgresSQL. PostGIS supports spatial data types and functions as described by the OpenGIS Simple Features Specification for SQL standard.
GeoTools
A Java library to perform manipulation of spatial data. GeoTools includes:
- Support for various spatial data formats (such as ESRI shapefile, Geography Markup Language (GML), MapInfo Interchange Format (MIF)), and spatial transformations.
- Java Topology Suite (JTS) from Vivid Solutions that defines Java classes to represent spatial data.

The example solution uses GeoTools for spatial transformations and JTS for converting the binary large object (BLOB) data from the database to a geometry.

OpenLayers
An open source JavaScript mapping library that lets you add dynamic maps to Web pages. You can overlay markers, lines, and polygons on top of base maps. The example in this article uses OpenLayers’ capability of adding a layer on top of a base map from a KML file.

Apache Tomcat
An open source J2EE JSP and servlet container used in the example to deploy the service. Download the sample code used in this article. See Resources for the software used in the example.

Getting ready
This section explains what you need to prepare to create the example application in this article.

Logical architecture
Figure 1 shows the logical architecture of the example solution.

Figure 1. Logical architecture

Boundary data
The U.S. Census Bureau uses the Topologically Integrated Geographic Encoding and Referencing (TIGER) format and database to describe and publish land attributes like roads, buildings, rivers, lakes, and census tracts. The TIGER database includes a dataset with the cartographic boundaries of states, counties, cities, and ZIP codes. This dataset is in ESRI shapefile format (see Resources). The example uses the state boundaries in the ESRI shapefile format.
Getting the data
The U.S Census Bureau provides a state boundary dataset in ARC/INFO Export (.e00), ARC/INFO Ungenerate, and ESRI shapefile (.shp) formats. This article uses the ESRI shapefile format. State boundaries are available for every state individually, and there's a shapefile that includes all state boundaries. The example uses the shapefile for all states (st99_d00_shp.zip).

Unzip the contents of the st99_d00_shp.zip file to get the st99_d00.shp shapefile and associated files (st99_d00.shx and st99_d00.dbf).

Importing the data into PostGIS
PostGIS does not provide a utility to directly import data in ESRI shapefile format into the database (as IBM® DB2® does). You have to follow a two-step process to import data into PostGIS:

1. PostGIS provides the shp2pgsql tool, which lets you convert ESRI shapefiles into SQL-suitable INSERT statements. To use this tool to create a file with SQL INSERT statements from the shapefile, use the command `shp2pgsql st99_d00.shp testdb.state_bounds > statebounds.sql`. This command, with no command-line options, will assume the default -1 value for SRS ID. This is fine for the purposes of this article, as you'll use GeoTools for reverse projections and not the database features for coordinate transformations.

2. After ensuring that the database is started, you can then run the `psql` command to run the script.

By default, the SQL file created by the shp2pgsql tool will contain table create and the statement to add the geometry column to the table. You can use various command-line options of the tool to not create the table. (See Resources for PostGIS documentation.)

Validating the import
You can validate the import process by using an admin tool like pgAdmin III or by executing SQL SELECT statements on the table. You can try PostGIS functions, such as `AsText()` and `AsBinary()`, in the geometry column to ensure that the import was successful.

The application
The application to create KML from the data you just imported will be a J2EE servlet. This servlet will:

- Receive the state FIPS code as input.
- Query the database.
- Perform coordinate transformation.
- Create a KML from the data.

Understanding the data
The data imported into the database can have more than one row for a given state FIPS code. There could be multiple non-contiguous tracts of land that might form a given state. For example, the state of Alaska comprises many islands. To keep our application simple, we'll select only the piece of land that has the greatest area.
**Querying the database and reading the geometry**

Geographical data can be represented in two formats: Well Known Binary (WKB) and Well Known Text (WKT). PostGIS provides `AsBinary()` and `AsText()` functions to convert the geometry column to WKB and WKT representations of the data, respectively. The SQL statement to query the data is in Listing 1.

**Listing 1. Query to retrieve the state boundary in WKB format**

```java
String stateBoundsSql = "SELECT AsBinary(the_geom), name FROM state_bounds WHERE state = " + stateCd + " ORDER BY area DESC LIMIT 1;"
```

The GeoTools library comes with the Java Topology Suite (JTS), which has the class `WKBReader` that converts WKB representations of geometry into Java objects that represent the geometry. You will use this to convert the state boundary data to a `MultiPolygon` Java object, which is another JTS class. Listing 2 shows an example.

**Listing 2. Read the geometry into a JTS Java object**

```java
WKBReader wkbRead = new WKBReader();
Geometry geom = wkbRead.read(resultSet.getBytes(1));
MultiPolygon multiPolygon = (MultiPolygon) geom;
```

**Coordinate transformations**

The data you imported from the U.S. Census Bureau is in a spatial reference system based on North American Datum 1983 (NAD83). It uses an ellipsoid approximation for the Earth. Web-based mapping libraries, such as OpenLayers, Google Maps, and Microsoft® Bing, are all based on the World Geodetic System (WGS) datum. To simplify the calculations involved in showing data on the Web-based map, these libraries assume a spheroid Earth, instead of an ellipsoid Earth. Thus, the data needs to be transformed before we can overlay on Web-based maps. You can use the transformation functions provided by the GeoTools library, as shown in Listing 3.

**Listing 3. Create transformation object**

```java
String SOURCE_WKT = "GEOGCS["GCS_North_American_1983"],DATUM["D_North_American_1983", SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433],AXIS["Latitude",NORTH]]";
String TARGET_WKT = "GEOGCS["WGS 84"],DATUM["World Geodetic System 1984",SPHEROID["WGS 84"],6378137.0,298.257223563, AUTHORITY["EPSG","7030"],AUTHORITY["EPSG","8901"],UNIT["degree",0.017453292519943295],AXIS["Geodetic latitude",NORTH],AXIS["Geodetic longitude",EAST],AUTHORITY["EPSG","4326"]]";

CRSFactory crsFactory = FactoryFinder.getCRSFactory(null);
CoordinateReferenceSystem sourceCRS = crsFactory.createFromWKT(SOURCE_WKT);
CoordinateReferenceSystem targetCRS = crsFactory.createFromWKT(TARGET_WKT);
Hints hint = new Hints(Hints.LENIENT_DATUM_SHIFT, true);
CoordinateOperationFactory coFactory = FactoryFinder.getCoordinateOperationFactory(hint);
CoordinateOperation co = coFactory.createOperation(sourceCRS,targetCRS);
MathTransform transform = co.getMathTransform();
```

Once you have the object to perform the coordinate transformations, you pass on a coordinate pair and transform the same, one at a time. Listing 4 shows an example.
Listing 4. Transform the coordinates

```
Coordinate[] coordinates = multiPolygon.getCoordinates();
for (int i = 0; i < coordinates.length; i++) {
    Coordinate coordinate = coordinates[i];
    double x = coordinate.x;
    double y = coordinate.y;
    DirectPosition point = new GeneralDirectPosition(x, y);
    point = transform.transform(point, point);
    double lat = point.getOrdinate(0);
    double lng = point.getOrdinate(1);
    ...
}
```

Creating the KML

Now that you have the transformed coordinates, all you need to do is use them to construct a KML document. (See Resources for KML documentation.) A KML document for representing polygon data has the layout shown below.

Listing 5. Sample KML for representing polygon data

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Document>
<Placemark>
<MultiGeometry>
<style>
<style>
<PolyStyle>
<color>66FF0000</color>
<colorMode>normal</colorMode>
<fill>1</fill>
<outline>0</outline>
</PolyStyle>
</Style>
<Polygon>
<LinearRing>
<coordinates>-109.045,36.99,0
  -109.0451,36.967,0
  -109.0452,36.968,0
  -109.0452,36.958,0
  -109.045,36.99,0
</coordinates>
</LinearRing>
</Polygon>
</MultiGeometry>
</Placemark>
</Document>
</kml>
```

To keep the code in this article very simple, we're not going to use an XML parser or JAXB to construct the KML document. You'll construct the KML by appending strings, as shown in Listing 6.
Listing 6. Create KML document

```java
String kmlHeader = "<?xml version="1.0" encoding="UTF-8"?>
   "xmlns="http://earth.google.com/kml/2.0""><Document><Placemark><MultiGeometry>
   <Style><PolyStyle>
     <color>66FF0000</color><colorMode>normal</colorMode><fill>1</fill>
     <outline>0</outline><PolyStyle></PolyStyle><Polygon><coordinates>

String kmlFooter = "</coordinates></LinearRing></Polygon></MultiGeometry></Placemark>
   </Document></kml>

String kmlBody = ";
for (int i = 0; i < coordinates.length; i++) {
   ..
   double lat = point.getOrdinate(0);
   double lng = point.getOrdinate(1);
   kmlBody += lng + "," + lat + "," + 0 + "\n";
}
```

Viewing the KML

The service to provide a KML document describing a state's boundary is ready. The next step is to create a Web page that will consume this service to display the boundary on a map using OpenLayers.

Listing 7. JavaScript to consume KML service to overlay the shape on the map

```javascript
map.addLayer(new OpenLayers.Layer.GML("KML",
   {format: OpenLayers.Format.KML,
    formatOptions: {
      extractStyles: true,
      extractAttributes: true,
      maxDepth: 2
    }
   }));
```

Figure 2 below shows Arizona's KML overlay using OpenLayer on top of a base map from Metacarta.
Figure 2. KML overlay on OpenLayers map

KML Layer

Demonstrates loading KML file on top of a basemap.

Conclusion

With mashups, enterprises can unlock the potential of all sorts of data. You can now provide users with the capability to create situational applications at a fraction of the cost of traditional application development paradigms.

Location intelligence mashups and map-based mashups are popular categories of mashups. With the advent of Web-based mapping providers, it's now easy to create map-based mashups with little or no capital investment. In this article, you learned how to use open source software to create a KML boundary service from an ESRI shapefile that can be used in mashups.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample code</td>
<td>os-kmiservice-SourceFiles.zip</td>
<td>11KB</td>
</tr>
</tbody>
</table>

© Copyright IBM Corporation 2010  
**Trademarks**  
(www.ibm.com/developerworks.ibm/trademarks/)