



TWaver[®] GIS for Java

Developer Guide

Jun 2011

Serva Software

info@servasoftware.com

<http://www.servasoftware.com>

PO Box 8143, Wichita Falls, Texas, USA 76307



For more information about Serva Software and TWaver please visit the web site at:

<http://www.servasoftware.com>

Or send e-mail to:

info@servasoftware.com

Jun, 2011

Notice:

This document contains proprietary information of Serva Software. Possession and use of this document shall be strictly in accordance with a license agreement between the user and Serva Software, and receipt or possession of this document does not convey any rights to reproduce or disclose its contents, or to manufacture, use, or sell anything it may describe. It may not be reproduced, disclosed, or used by others without specific written authorization of Serva Software.

TWaver, servasoft, Serva Software and the logo are registered trademarks of Serva Software. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S.A. and other countries. Other company, brand, or product names are trademarks or registered trademarks of their respective holders. The information contained in this document is subject to change without notice at the discretion of Serva Software.

Copyright © 2011 Serva Software LLC

All Rights Reserved

Table of Contents

- TWaver GIS for Java
 - Introduction
 - Dependencies
 - Structures
 - First Project
 - Using TWaver GIS
 - Using GeographyMap
 - Using GeographyLayer
 - Using GeographyFeature
 - Using Executor
 - Handle the events
 - MapEvent for map
 - MapLayerChangedEvent for layer
 - Combination
 - Little Widgets
 - Pre-defined toolbar
 - Pre-implemented operations
 - Using Navigator component
 - Using StatusBar component
 - Using WMSInformationTable
 - Using GisNetworkAdapter
 - Using InterceptedLink
- TWaver GIS FAQ
 - How to access map server via Http proxy
 - How to custom the cache of tile layer
 - How to synchronize the network view in map

TWaver GIS for Java

TWaver GIS for Java is an extended library of TWaver Java, built on the Java 2 platform. With the help of this library, developers can build Java applications be capable of presenting, operating maps, and even managing the geographic features. TWaver GIS also provides several predefined Swing components to help TWaver users manage or operate the map, such as the status bar, magnifier, and etc.

- [Introduction](#)
- [Using TWaver GIS](#)

Introduction

Under Java platform, there are many development kits to help programmers develop applications supporting the map. In order to reduce the development complexity for the users who have used TWaver, TWaver development team develops a new extent library named TWaver GIS for Java, to help TWaver users combine the network data with the geographic data. As this library is built with Swing technology, it can be deployed in an application or in an applet. With the help of the library, developers can access to some tile map servers and even the servers which provide WMS or WFS.

TWaver GIS supports equidistant projection and mercator projection.

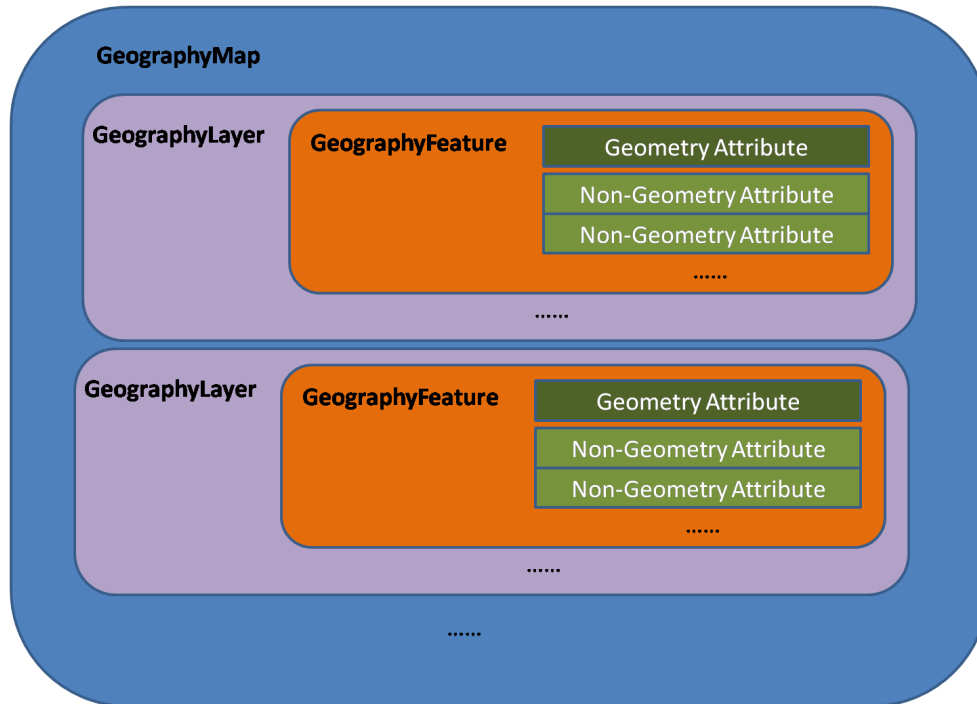
- [Dependencies](#)
- [Structures](#)
- [First Project](#)

Dependencies

TWaver GIS for Java needs twaver.jar, JDK/JRE 1.4 or later.

Structures

TWaver GIS defines 3 important interfaces to manage geographic data including GeographyMap, GeographyLayer and GeographyFeature. They are designed to present the geographic data and maintain their properties.



GeographyMap

This type is defined to organize geographic data integrally like a map we usually used. It consists of geographic layers. We can do operation on it such as zoom in, zoom out, pan, measure distance between geographic points and etc.

GeographyLayer

This type is defined to organize one kind of geographic data as a collection. It consists of geographic features. Accessing shape files directly, you can specify custom style to present the layer.

You can customize visible layers according to specified properties, e.g. the scale of the map. At the same time, you can get attributes value of any features by querying a layer.

GeographyFeature

This type is used to wrap the geographic data. It consists of the geometry attributes and non-geometry attributes of the geographic meta-data.

Executor

This type is used to define the engine to access geographic data. Different executors are distinguished by executor type. e.g. `TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS`, `TWaverGisConst.EXECUTOR_TYPE_OPENSTREET`, `TWaverGisConst.EXECUTOR_TYPE_ARCGIS_OGC` mean 3 kinds of engine, and they separately stand for the engines which can access the data provided by Geoserver Web Map Service, OpenStreetMap, and ArcGIS WMS.

First Project

Welcome to your first TWaver GIS project !

Getting Started

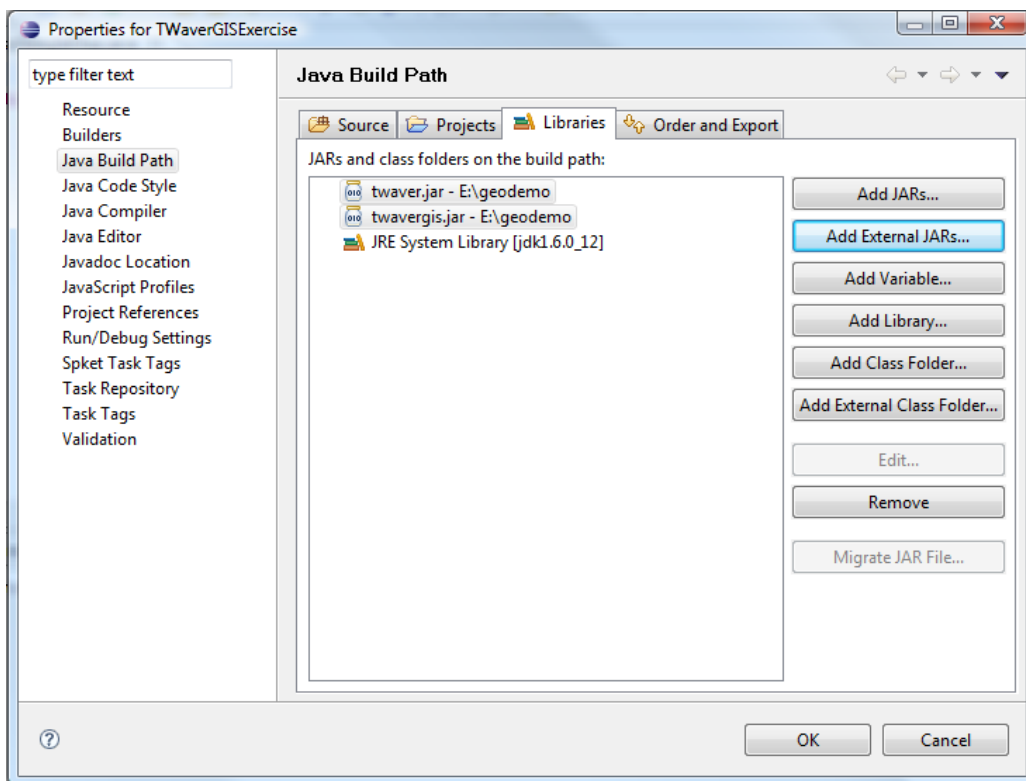
Ensure you have Java

Be sure you have got twaver.jar, twavergis.jar and Java installed.

Import library into project

If you use Eclipse as your IDE, you can follow below steps.

1. Start up Eclipse
2. Open up the File>New>Java Project menu, and create a Java project 'TWaverGISExercise'
3. Right click 'TWaverGISExercise' in the project view, start up the Properties menu, select Java Build Path>Libraries tab, and click Add External JARs button to import twaver.jar,twavergis.jar



As for other IDEs, make sure twaver.jar and twavergis.jar are imported as this project library.

Create your first class to get map from [OpenStreetMap](#).

```
import java.awt.BorderLayout;
import java.awt.Container;
```

```
import javax.swing.JFrame;
import twaver.GeoCoordinate;
import twaver.Link;
import twaver.Node;
import twaver.TDataBox;
import twaver.gis.GisNetworkAdapter;
import twaver.gis.TWaverGisConst;
import twaver.gis.gadget.StatusBar;
import twaver.network.TNetwork;

public class AccessMap {
    public static void accessMap() {
        JFrame frame = new JFrame("Access OpenStreetMap through TWaverGIS");
        TNetwork network = new TNetwork();
        GisNetworkAdapter adapter = new GisNetworkAdapter(network);
        adapter.installAdapter();
        adapter.getMap().addLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSMLAYER,
TWaverGisConst.EXECUTOR_TYPE_OPENSTREET);

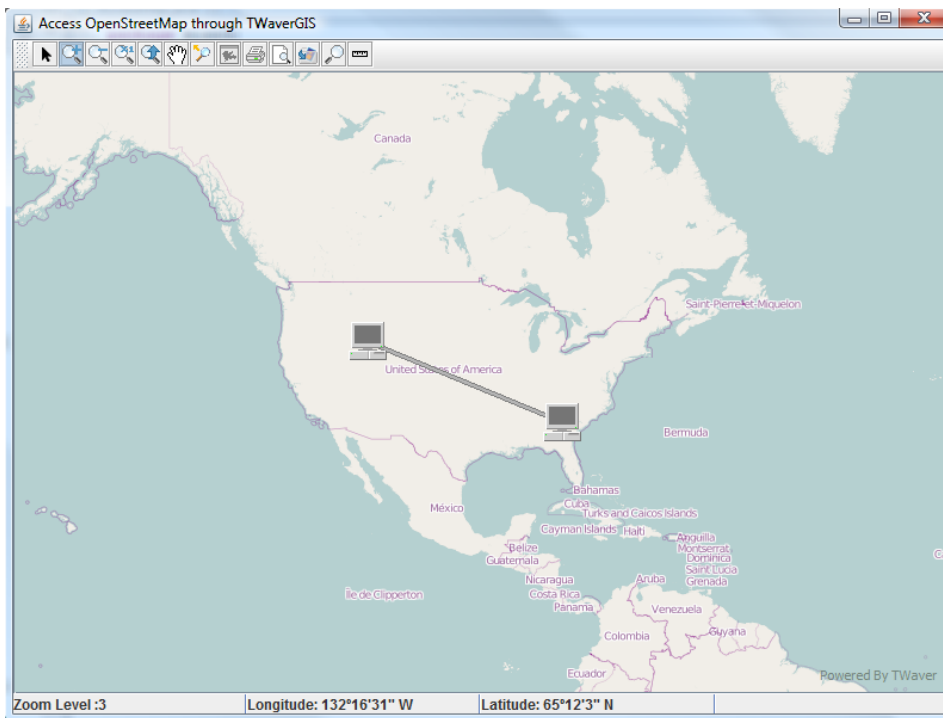
        TDataBox box = network.getDataBox();
        Node n1 = new Node();
        n1.putClientProperty(TWaverGisConst.GEOCOORDINATE, new GeoCoordinate(-111.6, 42.7));
        Node n2 = new Node();
        n2.putClientProperty(TWaverGisConst.GEOCOORDINATE, new GeoCoordinate(-82.9, 33.4));
        Link link = new Link(n1, n2);
        box.addElement(n1);
        box.addElement(n2);
        box.addElement(link);

        Container container = frame.getContentPane();
        container.setLayout(new BorderLayout());
        container.add(network, BorderLayout.CENTER);
        container.add(new StatusBar(adapter.getMap(), network.getCanvas()), BorderLayout.SOUTH);

        frame.setSize(800, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setLocationRelativeTo(null);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                AccessMap.accessMap();
            }
        });
    }
}
```

Then you will get the result like below screenshot :



Using TWaver GIS

One trend is the integration of GIS with other applications (both GIS and other IT systems) using service-oriented architectures (SOA). SOA can be used to integrate existing information systems to orchestrate work across those systems.

Another trend is the use of 2D and 3D geo visualization applications on the Web. Such as Google Map, OpenStreetMap, provide multi-resolution global basemaps with high performance and ease to use. Users value the ability to use the services as digital basemaps onto which they can layer their operational GIS information and tasks.

The Web Map Service (WMS) is an Open Geospatial Consortium (OGC) Web service standard for exchanging map information as map images (i.e., geo referenced pictures). The WMS specification also includes support for the use of Styled Layer Descriptors (SLD), which are used to specify a map's symbols and label properties.

The Open Geospatial Consortium's (OGC) Web Feature Service (WFS) is a standard protocol for serving geographic features across the Web. The GIS feature information that is encoded and transported using WFS includes both feature geometry and feature attribute values.

The feature information in WFS is encoded using GML which is used to express geographic information using XML. The Simple Features GML profile is recommended for use with ArcGIS and is the basis for WFS support in ArcGIS.

The basic Web Feature Service supports feature query and retrieval. A Transactional Web Feature Service (called WFS-T) enables the creation, deletion, and updating of features.

TWaver GIS for Java provides above two standard supports. With the support of WMS, TWaver GIS for Java requires the result images from map server, and according to the relative projection lays out those image as tiles. With the support of WFS, users of TWaver GIS can get the attributes of the specified geographical features, for examples the geom attributes and the other non-geom attributes.

- [Using GeographyMap](#)
- [Using GeographyLayer](#)
- [Using GeographyFeature](#)
- [Using Executor](#)
- [Handle the events](#)
- [Combination](#)

Using GeographyMap

GeographyMap is designed as a geographical layers container. It contains a series of geographic layers and presents them in specified order on a view component or an image.

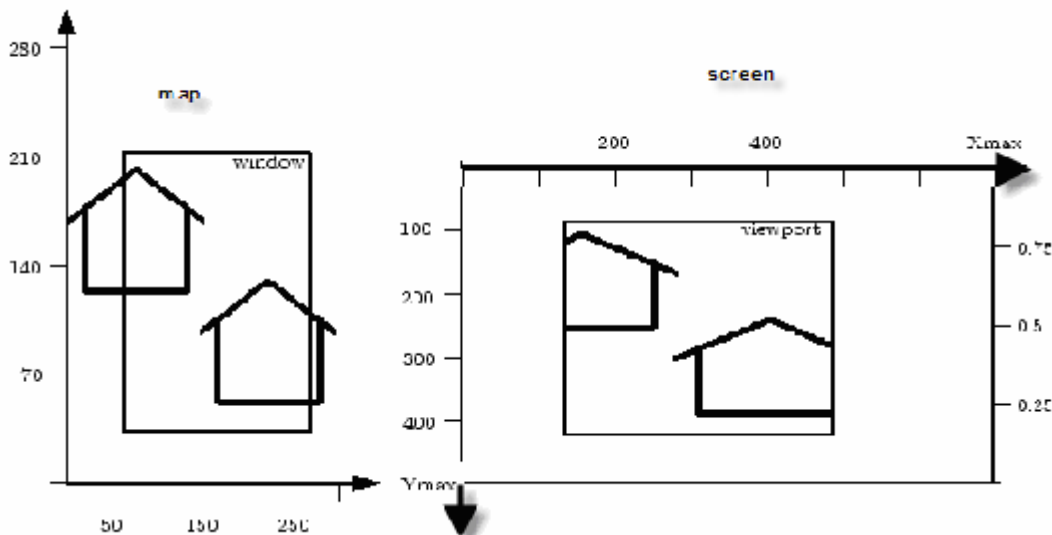
Create a map

If you want to use the tile mechanism to access the data from map servers, you should use the following code to create the other kind of map.

```
GeographyMap map = GisToolkits.createDefaultMap();
```

Present a map

TWaver GIS is mainly used to present geographic data. For presenting those datas, TWaver GIS draw a geographic map to the display devices. TWaver GIS defines its own viewport and window to do that.



From above figure, we know that if we want to display a map on a display device correctly, we should assign the volume of view port and volume of map window firstly.

```
//granted we have got a map object
//we assign a view port to the map. The view port is 800 pixels wide and 600 pixels height.
map.setViewport(new Rectangle(0,0,800,600));
//Then TWaver GIS will project the whole geographic data to a area (800 x 600) on
//the display device, such as a monitor, or an image.
//specify the center point of the map by the latitude and longitude coordinate.
//longitude is -73, and latitude is 34
map.setCenterPoint(new GeoCoordinate(-73,34));
map.setZoom(5);
//comments: By default, TWaver GIS assign the coordinate (0,0) as the center point, and level 0 as
//the default zoom level.
```

After the view port and the map window have been assigned correctly, programmers can invoke the following methods to draw the specified geographic data on the screen or an image.

```
BufferedImage image = new BufferedImage(800,600,BufferedImage.TYPE_INT_ARGB);
map.draw(image.createGraphics());
//or
map.draw(image);
```



For a TWaver user, if you are using TNetwork as view component to display topology data, you do not need to use above methods. TWaver GIS has wrapped them into [GisNetworkAdapter](#) which is able to set the map's view port automatically, present a map on TNetwork, and do operation on the map such as. panning, zooming, positioning network elements.

Operate a map

Manage layers

Add layers

As we mentioned in last chapter, map TWaver GIS is made up of geographical layers which contain the result images required from map servers. TWaver GIS defines an interface named [Executor](#) to provide the capability of accessing different map servers. The Executor can be regarded as map data engine. If you want to add a layer you should invoke method GeographyMap to add Layer.

```
addLayer(layerName,executorType);
//or
addLayer(layerName,executorType,serverURL);
```

For Example:

```
//if the data is provided by ArcGIS's WMS server
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_ARCGIS_OGC);
```

or

```
//if the data is provided by MapXtrem's WMS server
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_MAPINFO_OGC);
```

or

```
//if the data is provided by GeoServer
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS);
```

or

```
//if the data is provided by OpenStreetMap
map.addLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSMLAYER, TWaverGisConst.EXECUTOR_TYPE_OPENSTREET);
```

or

```
//add a layer which is required from a specified map server
map.addLayer("world",TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS,"http://www.serversoft.com/geoserver/wms?");
```



TWaver GIS implements several data engine to access data from map servers. TWaver GIS has the pre-defined URL of those servers. If programmers use a different URL, they will have to register the URL firstly.
For example:
The default URL of a server which supports [GeoServer](#) is "http://localhost:8080".
If the user builds a new server, and the URL changes into "http://geoserverpath:8080", programmers should reset the URL by invoking the following method in the application.

```
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_GEOSERVER,
"http://geoserverpath:8080"+TWaverGisConst.MAPDRIVER_GEOSERVER_POSTFIX);
```

Data sources supported by TWaver GIS:
Map services: WMS of GeoServer, ArcGIS, MapXtreme.
Tile services of OpenStreetMap,GoogleMap, etc.

After adding layers into map, TWaver GIS will manage the layers by their names or index numbers.

Remove layers

In TWaver GIS, developers can find a specified layer by its layer name or its index in the map. If you want to remove some layers from the map, invoke the method below:

```
map.removeLayer("ny");
```

or

```
map.removeLayer(0);
```

Move layers

GeographyMap presents the layers according to index. The layer with lower value will be drawn firstly. !gis_layers.png!You can invoke the method to move a layer upward/downward, also can move a layer to the top/bottom of the map.

```
map.moveLayer(layerName,layerMovedType);
```

or

```
map.moveLayer("ny", GeographyMap. LAYERMOVE_TYPE_DOWN);
```

or

```
map.moveLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSMLAYER, GeographyMap. LAYERMOVE_TYPE_TOTOP);
```


Using GeographyLayer

A geographic map is made up of many geographic layers. A layer object can access the specified map server and get the relative result images as tiles(each tile's size is 256*256).The layer will organize these tiles and lay out them according to the relevant projection.



All layers which are used to make up a geographical map should be presented with the same projection.

All tiles of the layer is rendered by the map server. TWaver GIS just gets the rendered result from the server. The style of the layer should be preset at the server side.

- Control the visibility of a layer
if a layer is added into GeographyMap, you can custom the layer visible or not and adjust the layer order.

```
layer.setVisibilityPermission(false);
```

- Using WMSInformationTable to read WMS capabilities.
Please refer to [Using WMSInformationTable](#)

Using GeographyFeature

TWaver GIS for Java defines Interface Geography Feature to describe geographical features. It can describe geometrical attributes and non-geometrical attributes of feature. TWaver GIS gets the map images via WMS or by requiring from tile servers, and TWaver GIS gets the information of geographical features via WFS. [Web Feature Service](#) interface is collection of operations (implemented as messages carried over HTTP) that allow a client to retrieve and manipulate geographic features. TWaver GIS implements the getCapabilities and getFeature interface, and is able to parse the WFS's capabilities, read the attributes of the features and redraw the shape of the features got from WFS on TNetwork component. TWaver GIS for Java provides those features by WFSUtils and WFSRequest classes.

Read the capabilities of WMS server

If you want to read the abilities of WFS server, invoke method WFSUtils.getWFSAbilities.

```
String contents = WFSUtils.getWFSAbilities(TWaverGisConst.EXECUTOR_TYPE_GEOSERVER, "http://customgeoserver/wfs?");
//or
//String contents = GisToolkits.getWFSAbilities(TWaverGisConst.EXECUTOR_TYPE_GEOSERVER, "http://customgeoserver/wfs?");
System.out.println(contents);
```

In some cases, developers just want to get the layers' information of WFS's server. Method getWFSLayers will be helpful.

```
List layers = WFSUtils.getWFSLayers(TWaverGisConst.EXECUTOR_TYPE_GEOSERVER, "http://customgeoserver/wfs?");
//or
//List layers = GisToolkits.getWFSLayers(TWaverGisConst.EXECUTOR_TYPE_GEOSERVER, "http://customgeoserver/wfs?");
if(wfsLayerInfos!=null){
    Iterator itearator = layers.iterator();
    while(itearator.hasNext()){
        WMSLayerInfo info = (WMSLayerInfo)itearator.next();
        System.out.println(info);
    }
}
```

Query features

When users interact with a map, they want to highlight some special geographical features or display the features information, or get the specified features by referring to some attribute. TWaver GIS provides two kinds of query operators to meet to the requirement, spatial query and comparison query.

Spatial query

A spatial operator determines whether its geometric arguments satisfy the stated spatial relationship. TWaver GIS will returns the features as a list if the spatial relationship is satisfied. Otherwise returns null.

```
//an rectangle area on the screen
Rectangle2D area = ...;
//a specified map object
GeographyMap map = ...;
String wfsUrl = "http://geoserver/wfs?";
String wfsLayer = "topp:state";
Point2D p = new Point2D.Double(area.getMinX(), area.getMaxY());
```

```
Point2D q = new Point2D.Double(area.getMaxX(), area.getMinY());
GeoCoordinate ul = GisToolkits.convertScreenToLatLong(map, p);
GeoCoordinate br = GisToolkits.convertScreenToLatLong(map, q);
GridBbox box = new GridBbox(ul.getLongitude(), ul.getLatitude(), br.getLongitude(), br.getLatitude());
GeographyFeature[] features = WFSRequest.requireFeatures(wfsUrl, WFSUtils.buildBBoxOperation(
    wfsLayer, new String[] {}, box));
if (features != null) {
    for (int i = 0; i < features.length; i++) {
        System.out.println(features[i]);
    }
}
```

With above segment, users will get the features that cover the rectangle area on the screen..

Comparison query

A comparison operator is used to require the features which satisfy the comparison between the feature's relative value and the reference value.

```
String wfsUrl = "http://geoserver/wfs?";
String wfsLayer = "topp:state";

ComparisonOperateCondition condition = new ComparisonOperateCondition(
    TWaverGisConst.COMPARISON_QUERY_BY_SINGLEOPERATOR);
String referenceProperty = "NAME";
condition.setReferenceProperties(new String[] {referenceProperty});
condition.setOperators(new int[] { TWaverGisConst.COMPARISON_OPERATOR_EQUAL });
String referenceValue = "New NY";
condition.setReferenceValues(new String[] { referenceValue});
GeographyFeature[] features = WFSRequest.requireFeatures(wfsUrl, WFSUtils.buildComparisonOperation(
    wfsLayer, new String[] {}, condition));
if (features != null) {
    for (int i = 0; i < features.length; i++) {
        System.out.println(features[i]);
    }
}
```

With above segment, users will get the features which 'NAME' attribute value is 'New NY'.

Using Executor

The Executor should be regarded as data engine. Each engine is designed to get the data from relative map server, and lay out the result tiles according to the relative projection.

For example:

EXECUTOR_TYPE_GEOSERVER represents the engine which can get the data from a GeoServer.

When add some geographic data into map, it is supposed to specify the engine type.

```
//supports Geoserver WMS
EXECUTOR_TYPE_GEOSERVER
//supports Geoserver WMS cache
EXECUTOR_TYPE_GEOSERVER_CACHE
//supports WMS of MapXtreme
EXECUTOR_TYPE_MAPINFO_OGC
//supports WMS of ArcGIS
EXECUTOR_TYPE_ARCGIS_OGC
//support OpenStreetMap tile server
EXECUTOR_TYPE_OPENSTREET
etc.
```

Above block lists all engines that TWaver GIS supports in real time.

Each engine should be assigned the URL of the relative map server. Before using the executor to access geographic data, you should specify the URL at first.

e.g.

```
//register the URL of a MapXtreme server's WMS
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_MAPINFO, "http://www.mapinfo.geoservice/wms?");
//register the URL of a ArcGIS server's WMS
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_ARCGIS, "http://arcgis/online/wms?");
////register the URL of a GeoServer's WMS
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_GEOSERVER, "http://customgeoserver/geoserver/wms?");
```



Because of some tile servers have stable URLs, users don't have to reset the URL of those servers which have been predefined in TWaver GIS. e.g. OpenStreetMap.

As above mentioned, if you want to add a layer got from custom MapXtreme server, use the following statements:

```
//reset the URL of the map server
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_MAPINFO, "http://www.mapinfo.geoservice/wms");
//add the layer with the relative Executor type.
map.addLayer("usa", TWaverGisConst.EXECUTOR_TYPE_MAPINFO_WMS);
```

If add a layer got from OpenStreetMap, you don't have to reset the URL of the map server, add the layer as below codes:

```
map.addLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSMLAYER, TWaverGisConst.EXECUTOR_TYPE_OPENSTREET);
```

In some cases, users need to add different layers from different servers. The following segment can be used.

```
map.addLayer("world",TWaverGisConst.EXECUTOR_TYPE_MAPINFO_WMS,"http://custommapinfo/mapservice");  
map.addLayer("topp:states",TWaverGisConst.EXECUTOR_TYPE_GEOSERVER,"http://customGeoServer/wms?");  
map.addLayer("ny",TWaverGisConst.EXECUTOR_TYPE_ARCGIS_OGC,"http://arcgis/wms?");
```

Handle the events

- [MapEvent](#) for map
- [MapLayerChangedEvent](#) for layer

MapEvent for map

Being operated, GeographyMap object will fire corresponding map events to indicate the finish. Any observers will receive that event and should do something relative.

MapEvent can be passed to every MapListener object which is registered to receive the "interesting" map events by map.addMapListener method. A MapEvent object is identified by event type which can be got by invoking MapEvent.getEventType method.

For example, if we set the center coordinates of a map, a map event with the type MAP_WINDOW_CHANGED will be generated.

```
map.addMapListener(new MapListener(){
    public void mapChanged(MapEvent evt){
        GeographyMap map = evt.getMap();
        int type = evt.getEventType();
        if(MapEvent.MAP_WINDOW_CHANGED == type){
            int zoom = map.getZoom();
            System.out.println("received map event:"+evt.getEventTypeDescription()+
                ", current zoom is "+zoom);
        }
    }
});
```

MapLayerChangedEvent for layer

When some layers of a map are removed, moved, set visible or invisible, the map object will generate some corresponding MapLayerChangedEvent objects. At the same time, the event will be passed to the registered event listener.

For example:

```
map.addMapLayerChangedListener(new MapLayerListener(){
    public void layerChanged(MapLayerChangedEvent evt){
        System.out.println(evt.getLayerName()+
            " ----- event type is "+evt.getEventTypeDescription());
    }
});
```


Combination

- [Little Widgets](#)
- [Using GisNetworkAdapter](#)
- [Using InterceptedLink](#)

Little Widgets

TWaver GIS provides some little view components to help users manage layers such as read the information of map services and etc. We call these components as gadgets.

- [Pre-defined toolbar](#)
- [Pre-implemented operations](#)
- [Using Navigator component](#)
- [Using StatusBar component](#)
- [Using WMSInformationTable](#)

Pre-defined toolbar

Generally, user need a convenient toolbar in their UI program. Continuing TWaver's tradition, TWaver GIS for Java provides predefined toolbar for developers.

If you have wrapped TNetwork by using method `GisNetworkAdapter` and invoking method `installAdapter`, TWaver GIS for Java will install a default toolbar on the TNetwork object.



If you are not familiar with how to use the toolbar of the TNetwork object, please refer to [Customing Toolbar](#) (TWaver Developer Guide).

```
GisNetworkAdapter adapter = new GisNetworkAdatper(network);
adapter.installAdapter();
network.setToolbarByName(TWaverGisConst.TOOLBAR_TILECLIENT);
```

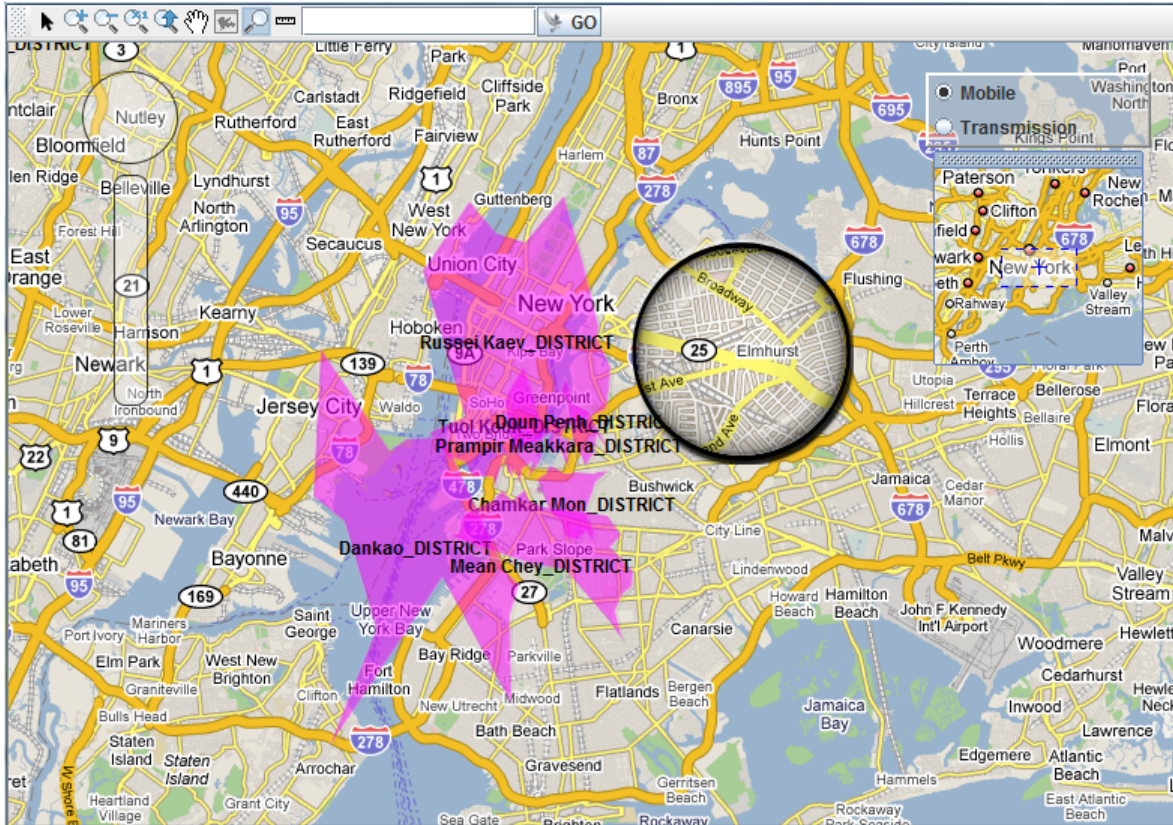
or

```
network.setToolbarByName(TWaverGisConst.TOOLBAR_GISSTAND);
```

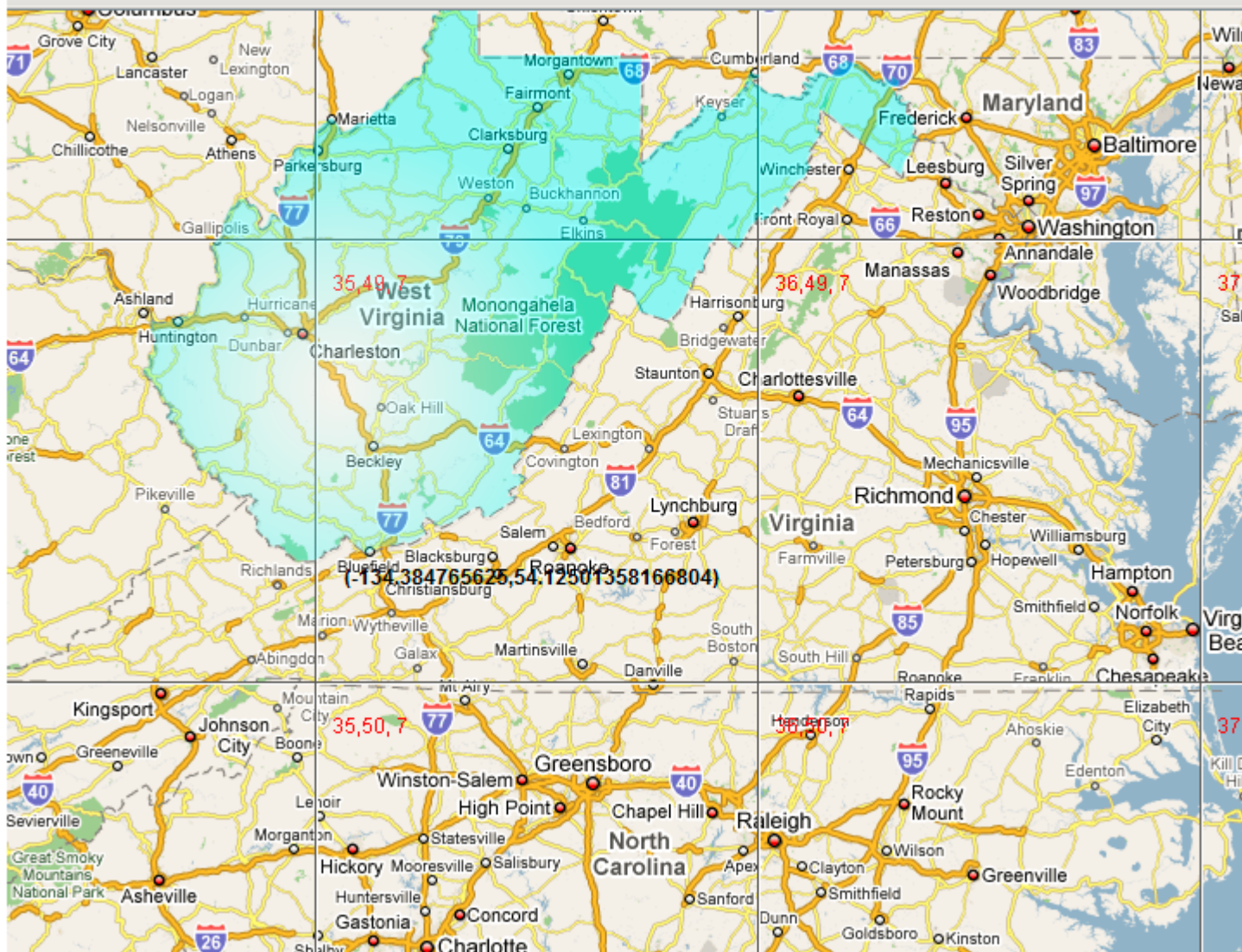


Pre-implemented operations

TWaver GIS pre-implements some general operations, like zooming in/out the map with mouse wheel or by double click.



Combine the data with different sources.



Using Navigator component

TWaver GIS provides a Navigator component to help user operate map object, like move right, move left, move up, move down, zoom in, zoom out.



```
Navigator navigator = new Navigator(network);  
navigator.showout(true);
```

If you want to hide the navigator when it is not active, you can set the dismiss delay value.

```
navigator.setDismissDelay(3000);
```

With above code, the navigator will hide after 3 seconds if it is not active.

Using StatusBar component

In an application, users always need to monitor some information of current view from a status bar.

TWaver GIS provides a pre-implemented StatusBar which can display the geographical coordinates of the mouse, display the current zoom level of the map and the distance information.

Scale : 1:24141900	Longitude: E115°25'24"	Latitude: N74°31'9"	Distance : 4644.27 km
--------------------	------------------------	---------------------	-----------------------

User can add a StatusBar component into their view component according to following code.

```
JPanel panel = new JPanel(new BorderLayout());
//network is a TNetwork instance
panel.add(network,BorderLayout.CENTER);
StatusBar statusbar = new StatusBar(map, network.getCanvas());
panel.add(statusbar,BorderLayout.SOUTH);
```


Using WMSInformationTable

WMSInformationTable is used to access WMS server. The table component can parse the WMS getCapabilities contents, and display all the layers information.

Layername	StyleName	SRS	minx	miny	maxx	maxy
tiger.poly_...		EPSG:4326	-74.047185	40.679648	-73.90782	40.882078
tiger.poi		EPSG:4326	-74.011831	40.707546	-74.001530	40.719885
tiger.tiger_r...		EPSG:900	-74.02722	40.684221	-73.907005	40.878178
sf.archsites						
sf.bugsites						
sf.restricted						
sf.roads		EPSG:26713	589434.85	4914006.3	609527.21	4928063.3
sf.streams		EPSG:26713	589434.49	4913947.3	609518.21	4928071.0
topp.tasma...	capitals	EPSG:4326	147.29100	-42.851001	147.29100	-42.851001
topp.tasma...	simple_roa...	EPSG:4326	145.19754	-43.423512	148.27298	-40.852802
topp.tasma...	green	EPSG:4326	143.83482	-43.648056	148.47914	-39.573891
topp.tasma...	cite_lakes	EPSG:4326	145.97161	-43.031944	147.219696	-41.775558
topp.states		EPSG:4326	-124.73142	24.955967	-66.969849	49.371735
tiger.giant...	giant_polyg...	EPSG:4326	-180.0	-90.0	180.0	90.0
serva:sh_bj	sh_bj_style	EPSG:4326	120.85288	30.6819477	121.93262	31.856515
serva:sh_r...	sh_road_st...	EPSG:4326	120.86453	30.693753	121.93349	31.842077
nurc:Arc_S...		EPSG:4326	-180.0	-90.0	180.0	90.0

With the help of the table, users can read the layers information, open a layer by popup menu. If you need to display this kind of table in the application, construct a table according to the following segment.

```
WMSInformationTable table = new WMSInformationTable("http://geoserver/wms?",
TWaverGisConst.EXECUTOR_TYPE_GEOSERVER,map);
```

The three parameters are separately the WMS server's url, the relative executor type, a relative GeographyMap object.

If the WMS is provided by GeoServer and the server has the tile cache ability, and developers want to get map images from the cache, developers should use the other constructor.

```
WMSInformationTable table = new WMSInformationTable("http://geoserver/wms?",
TWaverGisConst.EXECUTOR_TYPE_GEOSERVER,map,
"http://geoserver/tilecache");
```


Using GisNetworkAdapter

GisNetworkAdapter is designed as a wrapper. With the help of this wrapper, user can make TNetwork instance support GIS. After wrapped, the specified TNetwork instance can geographically layout topological elements according to their coordinates, request the relative geographic information of a specified coordinate pair, and present the geographic data as background.

For example:

```
GisNetworkAdapter adapter = new GisNetworkAdapter(network);  
adapter.installAdapter();
```

With above code, the specified network object will be able to access GoogleMap, and locate elements stored in the network databox at appropriate position of screen . At the same time, the network will get a pre-designed toolbar, which can help users to manage the map data.

When the GIS supports not required, User can remove it by invoking GisNetworkAdapter.unInstallAdapter method.

```
adapter.unInstallAdapter();
```

Using InterceptedLink

In some cases, TWaver users need to set link style to be 'TWaverConst.LINK_STYLE_DASH' in order to describe certain state of the link. TWaver will not be able to draw long link efficiently, because TWaver have to calculate the path of the whole long link. Users, who want to combine TNetwork component with the location service, will often meet this problem.

For example, you locate two network elements on different positions, which longitude/latitude coordinates respectively are (-73.12,34.01) and (120.11,33.98). There is a link between these two elements. If the map scale is changed to 1:10000 (a big scale), the link will be stretched very very long.

TWaver GIS provides a new kind of link named InterceptedLink to solve this kind of problem. No matter how long the InterceptedLink is, TWaver will just calculate its visible part on the screen. That can improve the performance greatly.

```
TDataBox box = network.getDataBox();
Node node = new Node();
box.addElement(node);
node.putClientProperty(TWaverGisConst.GEOCOORDINATE,
new GeoCoordinate(-73.12,34.01));
Node messageNode = new Node();
messageNode.putClientProperty(TWaverGisConst.GEOCOORDINATE,
new GeoCoordinate(120.11,33.98));
box.addElement(messageNode);
Link l = new InterceptedLink(node, messageNode);
l.putLinkStyle(TWaverConst.LINK_STYLE_DASH);
box.addElement(l);
```

TWaver GIS FAQ

- [How to access map server via Http proxy](#)
- [How to custom the cache of tile layer](#)
- [How to synchronize the network view in map](#)

How to access map server via Http proxy

If you want to get some geographic data on Internet via Http Proxy, you need the following block of statements to set up the Http Proxy.

```
Properties systemSettings = System.getProperties();
systemSettings.put( "proxySet", "true" );
systemSettings.put("http.proxyHost", "174.142.24.201");
systemSettings.put("http.proxyPort", "3128");
```

If you are using dynamic proxy or proxy configure file, using Applet should be a good solution.

How to custom the cache of tile layer

You can invoke the following method:

```
GisManager.registerDefaultSetting(TWaverGisConst.BUFFER_BLOCK_LIMIT, new Integer(100));
```

How to synchronize the network view in map
