

TWaver[®] GIS for Java

开发手册

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
 - TWaver GIS for Java入门
 - TWaver GIS for Java开发概述
 - TWaver GIS for Java架构
 - TWaver GIS for Java开发环境
 - 使用TWaver GIS for Java
 - 使用GeographyMap
 - 使用GeographyLayer
 - 使用GeographyFeature
 - 使用Executor
 - 事件处理
 - MapEvent
 - MapLayerChangedEvent
 - 整合
 - 使用小控件
 - 使用 WMSInformationTable
 - 使用导航条工具组件
 - 使用状态工具条组件
 - 预定义工具栏
 - 预实现的操作集
 - 使用GisNetworkAdapter
 - 使用InterceptedLink
- 与地图数据交互.

TWaver GIS for Java

TWaver GIS for Java是在TWaver Java上扩展出来的一个新包，它是在Java2的平台上编译的。TWaver GIS for Java可用于展示，操作地图，并且可以管理地理信息。TWaver GIS for Java也为用户提供了多个Swing组件，以便于用户更方便的操作地图，比如工具条，导航条，状态栏等。

- [TWaver GIS for Java入门](#)
- [使用TWaver GIS for Java](#)

TWaver GIS for Java入门

在Java平台下，有很多开发工具可以帮助程序员开发开发所支持的Map组件。为了减少TWaver用户的开发难度，TWaver开发组研发出一个新的产品分支TWaver GIS for Java来帮助解决Java框架下的拓扑数据与Map数据的整合问题。TWaver GIS for Java的技术为Swing，用户可以将其发布为应用程序或Applet。

引用TWaver GIS for Java开发包，用户可以访问WMS、WFS服务器以及部分tile服务器提供的Map数据。

TWaver GIS for Java支持墨卡托投影与等距投影。

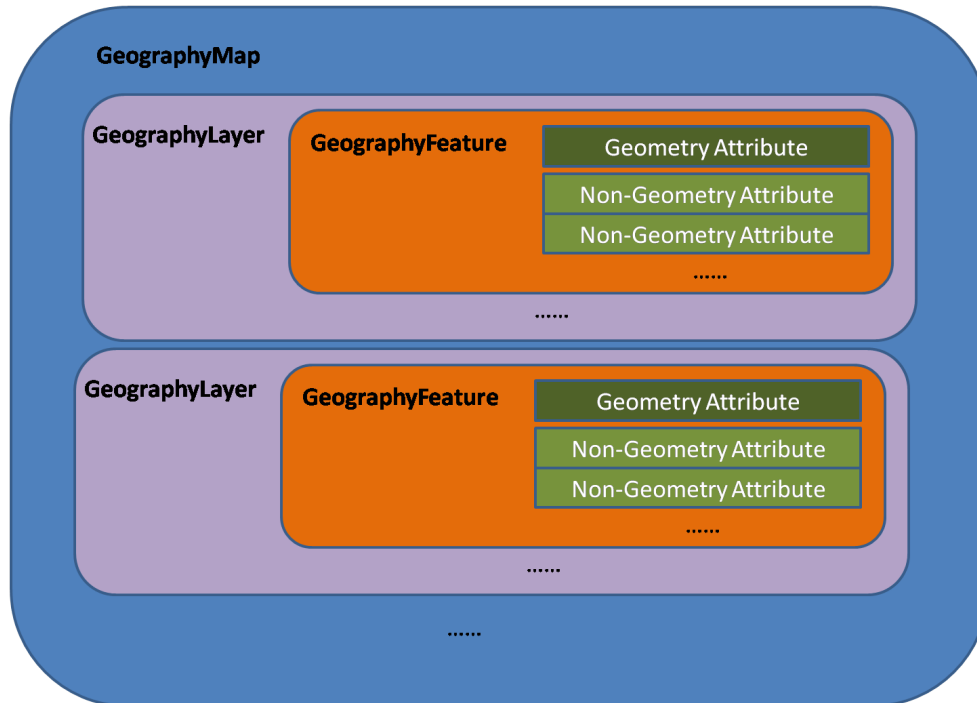
- [TWaver GIS for Java开发概述](#)
- [TWaver GIS for Java架构](#)
- [TWaver GIS for Java开发环境](#)

TWaver GIS for Java开发概述

TWaver GIS for Java的开发需要twaver.jar包，JDK/JRE 1.4+的版本。

TWaver GIS for Java架构

TWaver GIS for Java 定义了三种重要的接口来管理地理数据：GeographyMap, GeographyLayer 和 GeographyFeature. 它们用于展示地理数据并管理自身的属性。



GeographyMap

这种类型的接口用于定义组织地理数据，通常用来整合成一张地图。它是地理图层的集合，我们可以对其进行缩放，平移，多点测距等。

GeographyLayer

这种类型的接口用于定义组织成一种类型的地理数据，它是由地理要素组成的。通过直接访问地图文件，可以指定样式去渲染图层。

用户可以根据指定的属性定义可视图层。比如：地图的缩放级别。同时，也可以通过查询图层得到符合条件的任意地理要素属性值。

GeographyFeature


这种类型的接口定义用于地理数据。它是由地理数据中的几何属性和非几何属性组成。

Executor

它是用于定义访问地理数据的引擎。不同的executor定义类型也不同。比如TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS, TWaverGisConst.EXECUTOR_TYPE_OPENSTREET,

TWaverGisConst.EXECUTOR_TYPE_ARCGIS_OGC意为三种类型的引擎。它支持由Geoserver Web Map Service, OpenStreetMap和ArcGIS WMS引擎提供的数据库访问。

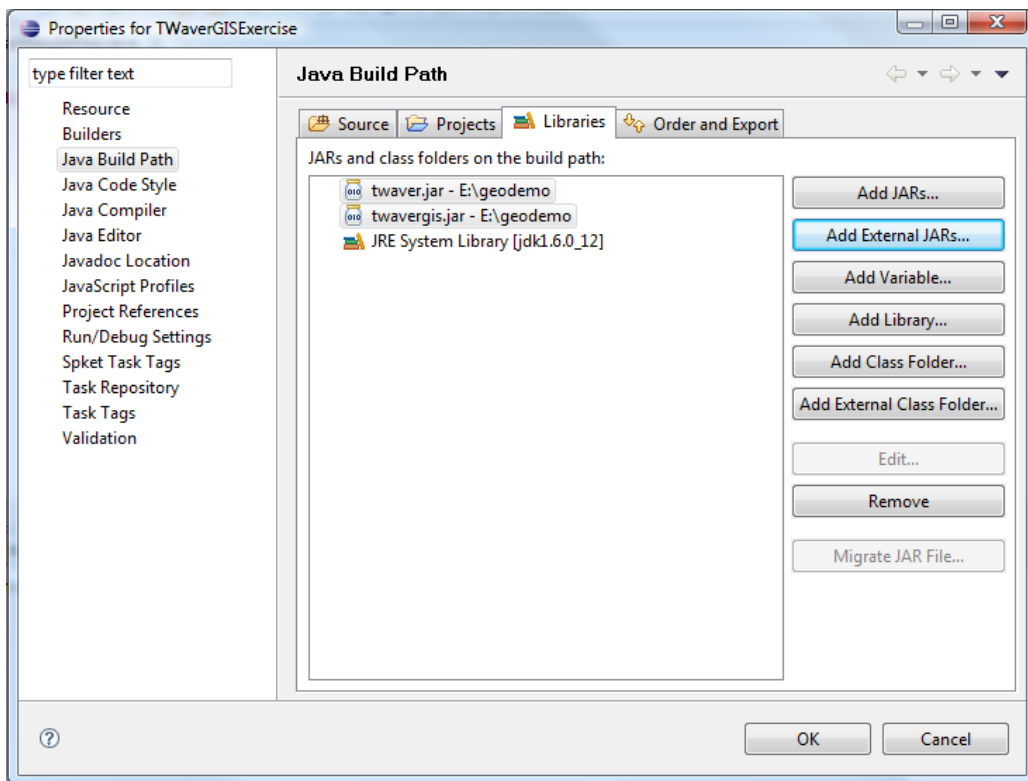
TWaver GIS for Java开发环境

 首先请确认您已经从[Serva Software](http://serva-software.com)官网申请到TWaver GIS for Java开发包。

引入GIS开发包

如果您使用的是Eclipse作为开发工具，可以参考下面的步骤进行配置

1. 启动Eclipse
2. 打开 File>New>Java Project 菜单，创建一个Java工程'TWaverGISExercise'
3. 右击 'TWaverGISExercise' 工程，在弹出的菜单中选择 Properties 子项，选择 Java Build Path>Libraries 选项卡，点击 Add External JARs 按钮导入 twaver.jar,twavergis.jar...



如果使用了其他的开发工具，请确保您将 twaver.jar 和twavergis.jar导入到工程中。

创建第一个例子---从OpenStreetMap中获取地图

```
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;
```

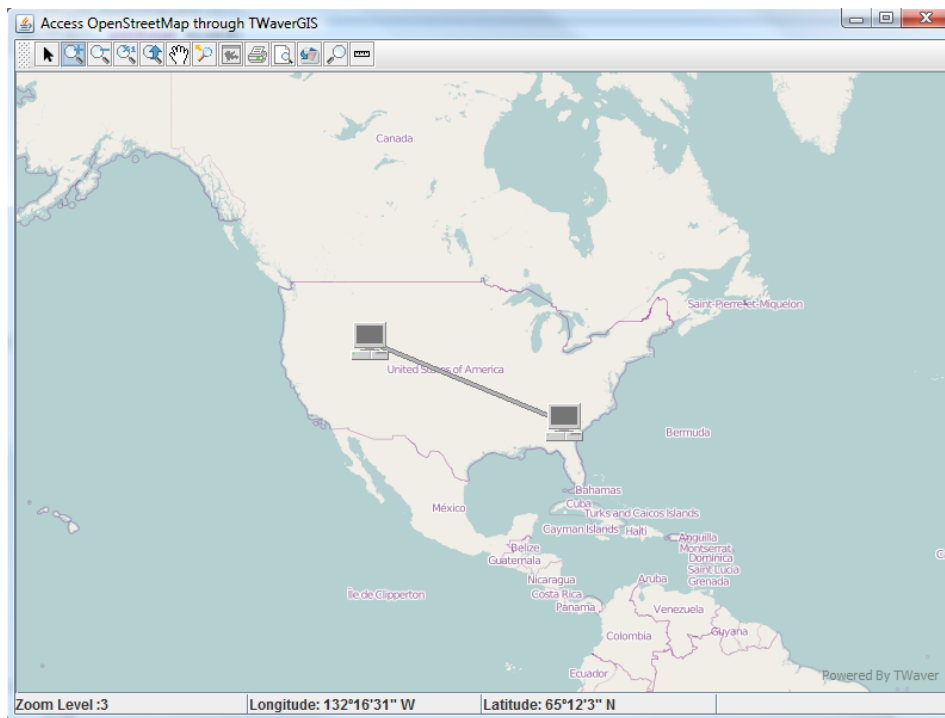
```
import twaver.gis.TWaverGisConst;
import twaver.gis.gadget.StatusBar;
import twaver.network.TNetwork;
public class AccessMap {
    public static void accessMap() {
        JFrame frame = new JFrame("Access OpenStreetMap through TWaverGIS");
        TNetwork network = new TNetwork();
        GisNetworkAdapter adapter = new GisNetworkAdapter(network);
        adapter.installAdapter();
        adapter.getMap().addLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSM_LAYER,
            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();
            }
        });
    }
}
```

运行上述代码，您将会得到如下所示的结果：



使用TWaver GIS for Java

不管是在GIS应用还是IT系统中，使用面向服务的体系结构(SOA)将GIS和其他的应用程序结合是现在的一种趋势。SOA可以将整合已存在的信息系统，以此协调工作。

另一种应用趋势是在Web中使用2D和3D地理信息。比如Google Map, OpenStreetMap，提供多种基于地图的，易用的，高性能的解决方案。

Web地图服务（WMS）是一种开放地理信息联盟（OGC）的Web服务标准，该服务利用具有地理空间位置信息的数据制作地图，其中将地图定义为地理数据可视的表现。Web地图服务所产生的地图一般是PNG, GIF或JPEG格式的，但偶尔也产生矢量格式的地图，如：SVG或WebCGM格式的地图。根据OGC规范，地图服务是专门提供共享地图数据的服务，负责根据客户程序的请求，提供地图图像、指定坐标点的要素信息、以及地图服务的功能说明信息。其实现需要利用HTTP和XML，对于传统的分布式计算技术，如CORBA、COM/DCOM和JavaBean等则不支持。相对于传统的分布式计算，HTTP和XML具有更强的适应性。在OGC地图服务中，利用HTTP的GET和POST两种方法实现客户端和Web Server的通信，在通信过程中主要依赖URL的参数配对来实现请求和交互响应。

开放地理信息联盟（OGC）的Web要素服务（WFS）是一种标准的Web地理要素服务协议。GIS地理要素是使用WFS编码和传输的，包括地理要素的几何图形和属性值。

WFS中的地理要素信息是使用GML进行编码的，GML使用XML来描述地理信息。

Web要素服务（WFS）返回的是要素级的GML编码，并提供对要素的增加、修改、删除等事务（WFS-T）操作，是对Web地图服务的进一步深入。OGC Web要素服务允许客户端从多个Web要素服务中取得使用地理标记语言（GML）编码的地理空间数据。

TWaver GIS for Java提供了以上两种标准的支持。在WMS的支持下，TWaver GIS从地图服务器上获取地图图片，并根据相关的投影来布局这些图片。在WFS的支持下，TWaver GIS的用户可以获取地理要素的属性，比如几何属性和非几何属性。

- [使用GeographyMap](#)
- [使用GeographyLayer](#)
- [使用GeographyFeature](#)
- [使用Executor](#)
- [事件处理](#)
- [整合](#)

使用GeographyMap

GeographyMap是一个地理图层的容器，它包含了很多图层，这些图层以一定的顺序叠加并显示在视图组件或图片上。

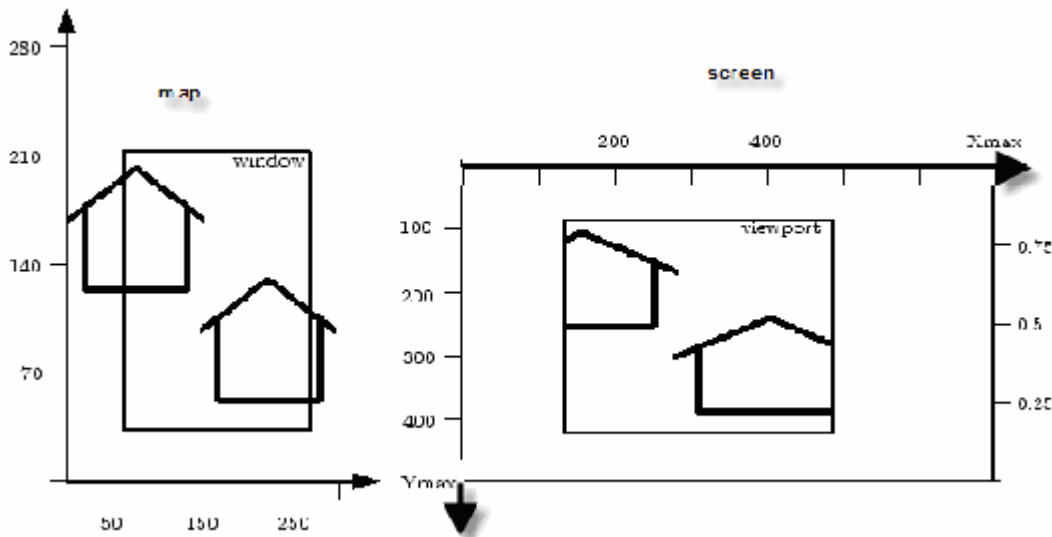
创建地图

如果您想要使用tile机制去访问地图服务器上的数据，可以使用下面的代码来创建地图。

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

展示地图

TWaver GIS for Java 主要用于展示地理数据。TWaver GIS for Java是使用地图去展示这些数据，它定义了自己的视口和窗口来实现这个。



从上图的配置中可以看到，我们如果想正确的展示设备，首先我们应该合理的分配视角窗口的大小。

```
//获取map对象
//给map对象设置视角窗口，视角窗口的大小为800×600
map.setViewport(new Rectangle(0,0,800,600));
//TWaver GIS for Java将会在800×600的视口中展示地理数据
//展示的设备可以为显示器或图片
//设置一个经纬度点作为地图的中心点
//经度为-73,纬度为34
map.setCenterPoint(new GeoCoordinate(-73,34));
map.setZoom(5);
//TWaver GIS for Java默认将（0，0）这个点作为地图的中心点，默认的缩放级别为0
```

当正确设置完视口和地图窗口之后，用户可调用下面的方法在屏幕或图片上绘制指定的地理信息：

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

```
map.draw(image);
```



对于TWaver开发者来说，如果你想要使用TNetwork作为视图组件去展示拓扑数据时，你完全没必要使用上面那种繁琐的方法。TWaver GIS for Java 使用自动获取地图视口的类GisNetworkAdapter来代替上面的方法将拓扑数据展示在地图上，并且还提供了一些交互，比如平移，缩放，设置网元的地理位置等。

操作地图

管理图层

添加图层

我们在上一章提及到TWaver GIS是由一系列包含图片的地理图层叠加而成，而这些图片是从地图服务器上获取。TWaver GIS for Java定义了一个名为Executor的接口来访问不同的地图服务器。可以把这个Executor看做是一个地图数据引擎。你可以通过调用GeographyMap类中的AddLayer方法来添加图层：

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

例如：

```
//如果地图数据是通过ArcGIS的WMS server提供的
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_ARCGIS_OGC);
```

或

```
//如果地图数据是通过MapXtrem的WMS server提供的
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_MAPINFO_OGC);
```

或

```
//如果地图数据是通过GeoServer提供的
map.addLayer("world", TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS);
```

或

```
//如果地图数据是通过OpenStreetMap提供的
map.addLayer(TWaverGisConst.TILEMAP_LAYERNAME_OSMLAYER, TWaverGisConst.EXECUTOR_TYPE_OPENSTREET);
```

或

```
//通过特定的地图服务器来提供图层
```

```
map.addLayer("world",TWaverGisConst.EXECUTOR_TYPE_GEOSERVER_WMS,"http://www.serversoft.com/geoserver/wms?");
```



TWaver GIS for Java提供了多种不同的数据引擎来访问地图服务数据。
TWaver GIS for Java预定义了如下的这个服务的URL，如果程序员想要更改URL，需要先注册一下URL。
比如：
服务器默认的URL支持GeoServer："http://localhost:8080".
如果用户新建了一个服务器，想要把URL更改为"http://geoserverpath:8080"，必须先调用下面的方法来恢复URL。

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

TWaver GIS for Java的数据源支持如下几种:
地理服务: WMS of GeoServer, ArcGIS, MapXtreme.
分片服务: OpenStreetMap, GoogleMap, etc.

map上添加完图层后，TWaver GIS for Java通过图层的名字或索引号来管理。

删除图层

在TWaver GIS for Java中，开发者通过图层的名字或索引号来管理，如果想要删除某个图层，可通过下面方法来处理：

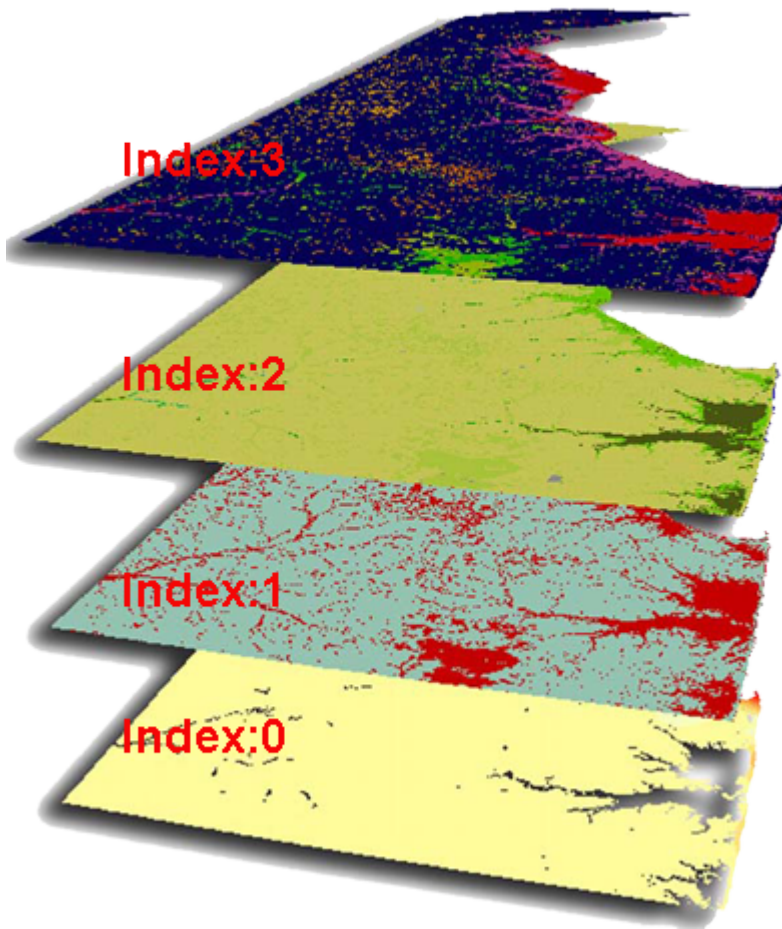
```
map.removeLayer("ny");
```

或者

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

移动图层

GeographyMap是通过索引号来绘制图层的，索引号小的图层会先绘制，因此会显示在最底层：



你可以调用下面方法来上移，下移图层：

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

或


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

或

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


使用GeographyLayer

地图是有很多张地理图层组成的。一个图层对象可以访问指定的地图服务器并从中获取相关的图片结果集作为tiles (每一个tile的大小为256*256)。然后图层会根据相关的投影来组织布局这些tiles。

 组成地图的所有图层应该保持相同的投影。

图层上所有的切片都是从地图服务器上渲染而成的，TWaver GIS for Java从服务器上去获取渲染的结果。图层的样式也是在服务端展示。

- 控制图层是否可见
当一个图层添加到地图中后，你可以定制这个图层是否可见，也可以调整图层的顺序。

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

- 使用WMSInformationTable读取WMS的性能
请参见[Using WMSInformationTable](#)

使用GeographyFeature

TWaver GIS for Java通过GeographyFeature接口来描述地理要素。可用于描述几何要素和非几何要素。TWaver GIS for Java 通过WMS或切片服务器来获取地理图片。TWaver GIS for Java通过WFS获取地理要素的信息。[Web Feature Service \(WFS \)](#) 是一个操作集合，它允许客户端重新获取或操作地理要素。TWaver GIS for Java实现了getCapabilities和getFeature接口，去获取WFS的服务级元数据，读取地理要素的属性，在Network组件上重画地理要素的形状。

TWaver GIS for Java通过WFSUtils和WFSRequest两个类来提供这些地理要素。

读取WMS服务器的性能

用户可以通过调用WFSUtils.getWFSAbilities来读取WFS服务器的性能。

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

有时，用户想获取WFS服务器上的地图图层的信息，可以通过getWFSLayers方法实现。

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

查询地理要素

当用户和地图交互时，有时可能需要高亮某些特定的地理要素，或展示特定地理要素的信息，或者通过一些条件属性来查询相关的地理要素。TWaver GIS for Java针对这种需求提供了两种解决方案：空间查询和比较查询。

空间查询

空间查询的用户需要确定哪些地理要素符合他指定的几何图形，TWaver GIS for Java会根据用户给定的几何图形返回所有符合条件的地理要素的集合，否则返回空。

```
//在屏幕上定义一个矩形区域
Rectangle2D area = ...;
//定义一个地图对象
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]);
    }
}
```

上面的代码片段将返回当前屏幕上矩形区域中包含的所有地理要素。

比较查询

比较查询指用户通过地理要素的某个属性值比较来查询符合条件的地理要素。

```
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]);
    }
}
```

以上的代码片段是获取'NAME'为'New NY'的地理要素。

使用Executor

Executor可以看作是一个数据引擎。每一个数据引擎是从相关的地图服务器上获取数据并根据投影来组织这些tiles的布局

例如：

EXECUTOR_TYPE_GEOSERVER是一个从GeoServer上获取数据的引擎。

当用户往地图中添加一些地理数据时，需要指定引擎的类型

```
//支持Geoserver WMS
EXECUTOR_TYPE_GEOSERVER
//支持Geoserver WMS cache
EXECUTOR_TYPE_GEOSERVER_CACHE
//支持WMS of MapXtreme
EXECUTOR_TYPE_MAPINFO_OGC
//支持WMS of ArcGIS
EXECUTOR_TYPE_ARCGIS_OGC
//支持OpenStreetMap tile server
EXECUTOR_TYPE_OPENSTREET
```

等等...

以上代码中列出了TWaver GIS for Java支持的所有实时的数据引擎。

每一个数据引擎都需要制定一个地图服务器。在使用executor去访问数据之前，需要先指定URL。

例如：

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



由于某些切片地图数据服务器已经有比较固定的URL了，是TWaver GIS for Java中预定义好的，用户无需再定义这些服务器的URL，比如OpenStreetMap。

从上面的介绍中可知，如果想要在MapXtreme服务器上添加一个图层，可以通过下面代码实现：

```
//设置地图服务器的URL
GisManager.registerDefaultSetting(TWaverGisConst.MAPDRIVER_MAPINFO, "http://www.mapinfo.geoservice/wms");
//通过数据引擎添加一个图层
map.addLayer("usa", TWaverGisConst.EXECUTOR_TYPE_MAPINFO_WMS);
```

如果想要添加一个OpenStreetMap的图层，你无需重设地图服务器的URL，而只需通过下面代码添加一层：

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

有时用户想添加不同服务器上的图层，可通过下面代码实现：

```
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?");
```

事件处理

- [MapEvent](#)
- [MapLayerChangedEvent](#)

MapEvent

在操作地图之前，GeographyMap 将会派发出一个地图事件来表明地图已经加载完成。任何观察者都可以监听到这个事情并进行自己的一些相关的操作。

每一个通过map.addListener注册的用于接收相关的Map事情的MapListener都会传递过来一个MapEvent对象，MapEvent对象可以通过调用MapEvent.getEventType来获取事件类型。

例如，当我们设置地图的中心点经纬度时，将会派发出一个MapEven.MAP_WINDOW_CHANGED类型的事件。

```
map.addListener(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

当地图图层被删除、移动、设置可见或不可见时，地图对象将会派发出相应的MapLayerChangedEvent事件。同时，事件将会被传到注册的事件监听器中。

例如：

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


整合

- [使用小控件](#)
- [使用GisNetworkAdapter](#)
- [使用InterceptedLink](#)

使用小控件

TWaver GIS for Java 提供了一些小的视图组件用于帮助用户管理图层、读取地图服务器的信息等。我们称这些组件为小控件。

- [使用 WMSInformationTable](#)
- [使用导航条工具组件](#)
- [使用状态工具条组件](#)
- [预定义工具栏](#)
- [预实现的操作集](#)

使用 WMSInformationTable

WMSInformationTable是用于访问WMS服务器。这个表格组件可以获取WMS上所有的内容，并展示所有图层的信息。

Layername	StyleName	SRS	minx	miny	maxx	maxy	
tiger:poly_l...		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			3...	4914490.8...	608346.46...	4926501.8...	
sf.bugsites			3...	4914107.8...	608462.46...	4920523.8...	
sf.restricted			8...	4916236.6...	599648.92...	4925872.1...	
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_f...	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	▼

用户可以通过这个表格读取到所有图层的信息，并通过右键菜单打开一个图层。

如果要在你的应用中添加这种表格，可以通过下面方法再建立这样的表格：

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

这个方法中WMS服务器URL，地图数据executor，GeographyMap这三个参数是分离的。

如果WMS是通过GeoServer提供的，并且服务器有缓存切片的能力，开发者可以通过下面的方法得到缓存的图片。

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

使用导航条工具组件

TWaver GIS for Java 提供了一个导航条组件用于帮助用户操作地图对象，比如右移，左移，上移，下移，放大，缩小。



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

如果想要在不活动时隐藏导航条，可通过设置隐藏的时间：

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

上面代码设置了当界面3秒处于不激活状态将隐藏导航条。

使用状态工具条组件

在某些应用中，用户总是需要通过状态工具条来实时监控当前视图上的一些信息。

TWaver GIS for Java提供了预定义的状态栏来显示当前鼠标的经纬度、当前的缩放级别以及距离。

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

用户可以通过下面代码将状态工具条加入地图中。

```
JPanel panel = new JPanel(new BorderLayout());
//network是TNetwork的一个实例
panel.add(network,BorderLayout.CENTER);
StatusBar statusbar = new StatusBar(map, network.getCanvas());
panel.add(statusbar,BorderLayout.SOUTH);
```

预定义工具栏

通常来说，用户在开发UI界面时都需要一个比较方便的工具条。承接TWaver的传统，TWaver GIS for Java也为开发者提供了一个预定义的工具条。

当通过GisNetworkAdapter中的installAdapter为network安装adapter时，TWaver GIS for Java在network上将会按装一个默认的工具条。



如果你不熟悉TNetwork上的工具条，可以参见TWaver开发手册中的定制工具条章节。

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

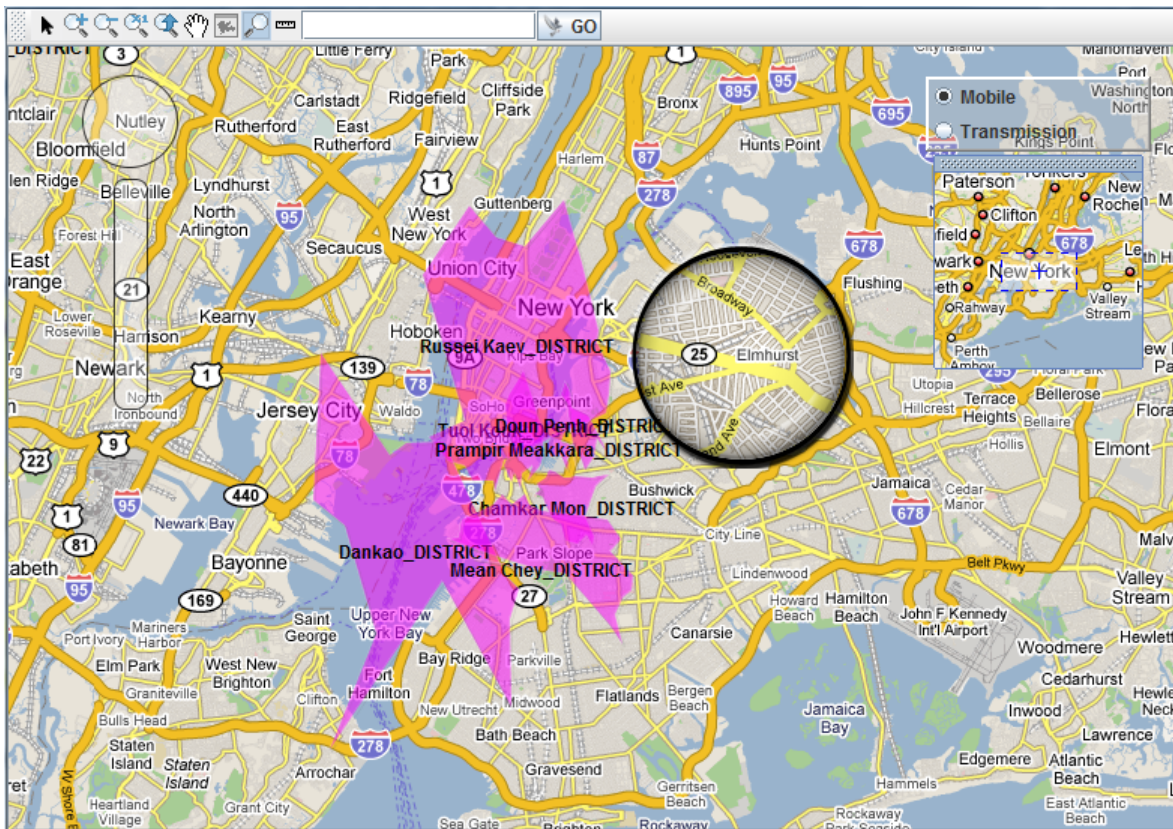
或

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

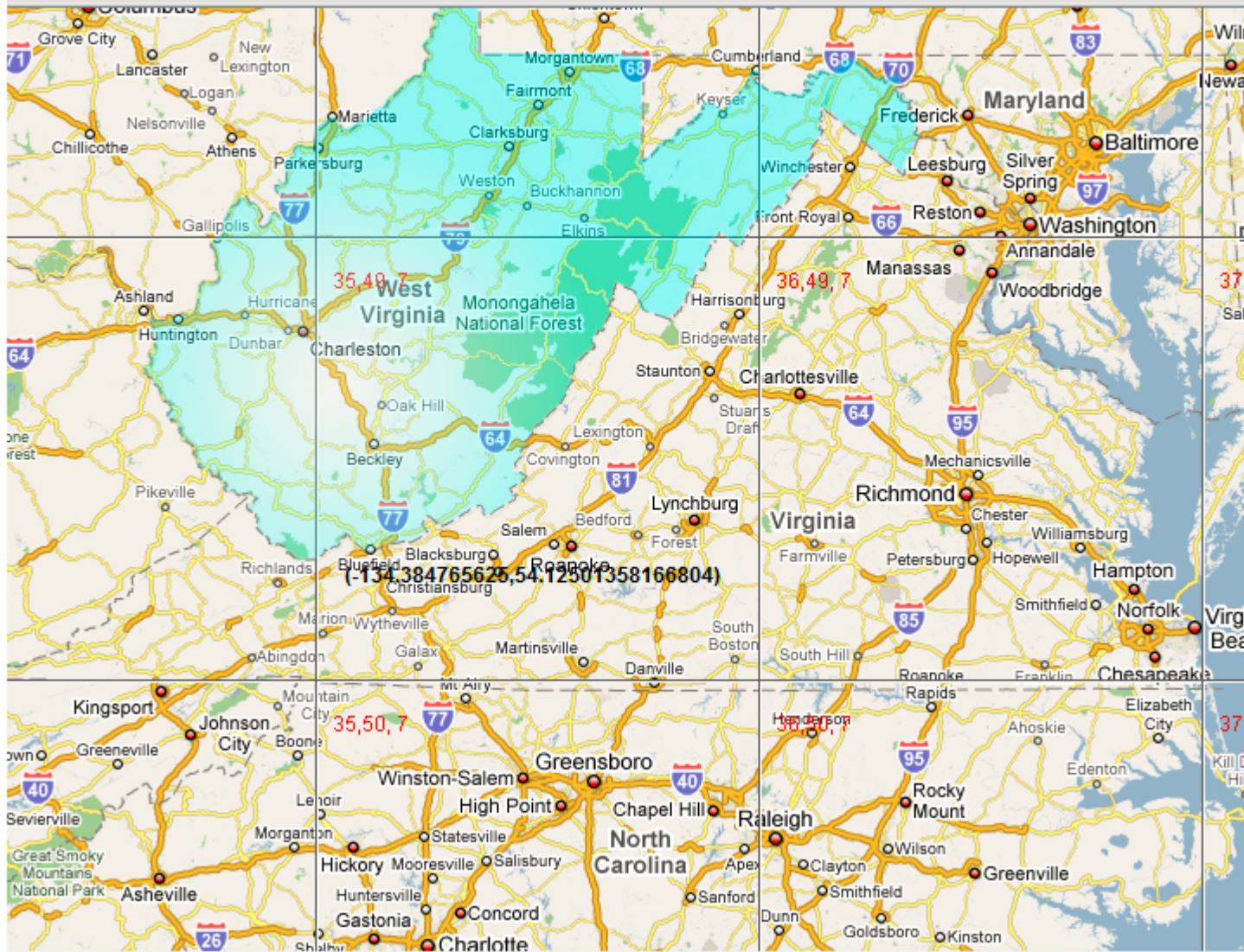


预实现的操作集

TWaver GIS for Java预定义了一些基本的操作集，比如通过鼠标滚轮或双击放大，缩小。



整合不同数据源的数据。



使用GisNetworkAdapter

GisNetworkAdapter可以看成是一个包装纸，在它的帮助下，用户可以使用支持GIS的TNetwork对象。在GisNetworkAdapter的包裹下，TNetwork实例可以根据网元的经纬度，相关经纬度的地理信息将其定位到实际的地理坐标中，并以背景的形式展示这些地理数据。

例如：

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

通过上面的代码，指定的network对象可以访问支持GoogleMap的network上合适位置的网元。同时，network也可以得到预定义的工具条去帮助用户来管理地图数据。

当不需要GIS时，用户可以调用GisNetworkAdapter.unInstallAdapter将其卸载。

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

使用InterceptedLink

某些情况下，TWaver的用户设置了Link的样式TWaverConst.LINK_STYLE_DASH来表示某种link的状态时，TWaver却不能很高效的去绘制出这些link，因为TWaver需要计算整个Link的长度并绘制出来。

和Network组件整合的用户通常会遇到这样的问题。比如，在network上设置两个不同位置的网元，它们的经纬度分别为(-73.12,34.01) 和 (120.11,33.98)，这两个网元之间有一条link，当地图缩放到1：100000（非常大）的级别时，这时Link将会延伸的非常非常的长。

为了解决这一问题，TWaver GIS for Java为用户提供了一种名为InterceptedLink。不管InterceptedLink有多长，TWaver只会计算当前屏幕可见的部分，这样就能很大的解决link的性能问题。

```
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);
```

与地图数据交互.

如上章节所述，当应用程序中有大量的网元数据需要呈现时，应当将这些网元数据以地理要素形式放置在空间数据库中，TWaver GIS负责将这些地理要素以网元形式显示在界面中，这样当用户需要与网元交互时，

TWaver GIS通过查询地理要素，然后将这些地理要素以网元的形式显示出来。

例如，一个资源管理系统需要维护数千计的光纤，每个光纤又被分成N段，这样在显示这些 数据时，因为数据量过大，会出现性能问题，此时以地理要素形式来显示网元数据将是最佳解决方案。

下面来详细解释如何将网元数据放置在空间数据库中，以及如何与相应的地理要素进行交互。

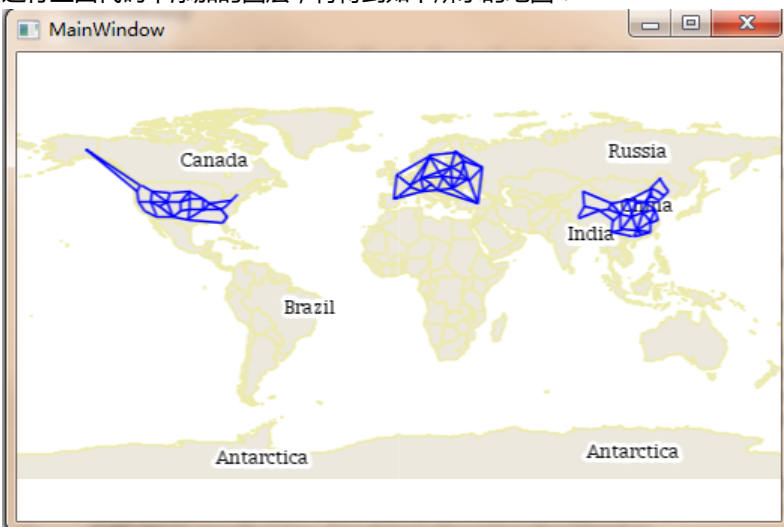
组合Map与Network组件

请参照[Adapter](#)章节进行设置.

```
private void InitMap()
{
    map.AddLayer("serva:world", GISConsts.EXECUTOR_TYPE_GEOSERVER_WMS_4326,DemoConsts.wmsServer );

    map.AddLayer(DemoConsts.FIBERTEMPLATELAYER,GISConsts.EXECUTOR_TYPE_GEOSERVER_WMS_4326,DemoConsts.wmsServer);
}
```

运行上面代码中添加的图层，将得到如下所示的地图：



显示网元对象的地理要素

如果想在地图上展示某些地理要素时，可以从空间数据库中得到地理要素并通过twaver的网元展示。

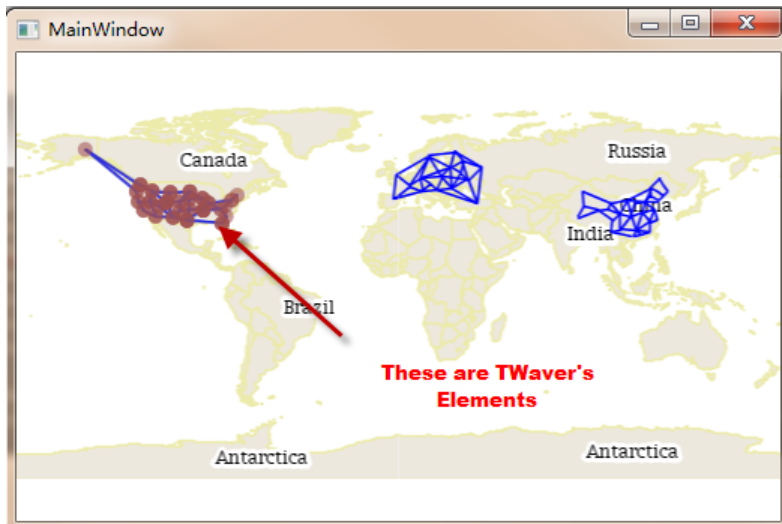
```
private void FillDataWithFeatures(List<Feature> features)
{
    FillNetwork(network, features);
}
private void InitNetwork()
{
    network = new Network();
    Adapter adapter = new Adapter();
    adapter.BindNetworkWithTWaverMap(map, network);
    BBoxQuery(DemoConsts.FIBERTEMPLATELAYER, new GeoCoordinate(10, -150),
        new GeoCoordinate(90, -60), FillDataWithFeatures);
}
```

```

    }
    public static void FillNetwork(Network network, List<Feature> features)
    {
        FeatureConverter converter = new FeatureConverter();
        network.ElementBox.Clear();
        foreach (Feature feature in features)
        {
            Object result = converter.ToTarget(feature);
            if (result is Element)
            {
                Element element = result as Element;
                if (element != null)
                {
                    network.ElementBox.Add(element);
                }
            }
            else if (result is List<Node>)
            {
                List<Node> nodes = result as List<Node>;
                foreach(Node node in nodes)
                {
                    network.ElementBox.Add(node);
                }
            }
        }
    }
    public static void BBoxQuery(String layerName, GeoCoordinate bl, GeoCoordinate ur, FeaturesHandler handler)
    {
        List<GeoCoordinate> bounds = new List<GeoCoordinate>();
        bounds.Add(bl);
        bounds.Add(ur);
        Geom geom = new EnvelopeGeom(bounds);
        SpatialCondition condition = new SpatialCondition(geom);
        condition.OperatorType = (GISConsts.SPATIAL_OPERATOR_TYPE_BBOX);
        QueryAction action = new QueryAction(DemoConsts.SERVA_NAMESPACE, layerName, condition);
        #if SILVERLIGHT
            WFSUtils.Query(handler, DemoConsts.wfsServer,
                WFSUtils.buildQueryOperator(DemoConsts.SERVA_NAMESPACE, new TWaver.GIS.OGC.Actions.Action[]
                { action }));
        #else
            List<Feature> features = WFSUtils.Query(DemoConsts.wfsServer,
                WFSUtils.buildQueryOperator(DemoConsts.SERVA_NAMESPACE, new TWaver.GIS.OGC.Actions.Action[] {action}));
            if (handler != null)
            {
                handler(features);
            }
        #endif
    }
}

```

运行上述代码，可得到下图



附加：

```
public class DemoConsts
{
    //public const
    public static String server = "http://twaver.servasoft.com:8000/geoserver/wfs?";
    public static String toppSpace = "topp=\"http://www.openplans.org/topp\"";
    public static String usLayer = "topp:states";
    public static String SERVA_NAMESPACE = "serva=\"http://www.servasoftware.com/gis\"";
    public static String USONWORLD = "usonworld";
        public static String TERMINALLAYER = "serva:twpoint";
        public static String FIBERTEMPLATELAYER = "serva:twline";
        public static String EDITLAYER = "serva:twLineString";

    public static String wfsServer = "http://twaver.servasoft.com:8000/geoserver/wfs?";
    public static String wmsServer = "http://twaver.servasoft.com:8000/geoserver/wms?";
}
```