【GIS系列】GeoTools简介及工具类分享

> 本文将对GeoTools相关概念进行介绍，同时会给大家分享我工作中用到的工具类及使用方法。
>
>
> **作者：后端小肥肠**

1.前言

GeoTools是一个功能强大的开源GIS工具库，为地理空间数据处理和分析提供了丰富的功能和便利的接口。无论您是GIS专业人士还是对地理空间数据感兴趣的开发人员，掌握GeoTools都是必不可少的。本文将从基本概念入手，介绍GeoTools的功能和使用方法，并重点分享一些实用的工具类和技巧，希望能为您在GIS应用开发中提供帮助和启发。

2. GeoTools简介

GeoTools是一个开源的Java库，用于处理和分析地理空间数据。它提供了一系列用于读取、写入、处理和可视化地理空间数据的工具和API。以下是与GeoTools相关的一些重要概念：

1. **地理空间数据（Geospatial Data）：** GeoTools主要用于处理地理空间数据，这包括地图、地形、卫星影像、矢量数据等地理信息。这些数据通常具有地理坐标信息和地理属性信息。
2. **地理坐标系统（Geographic Coordinate System，GCS）：** 地理坐标系统是用于在地球上定位点的一种方法。GeoTools支持多种地理坐标系统，包括经纬度坐标系统等。
3. **投影坐标系统（Projected Coordinate System，PCS）：** 投影坐标系统是将地球表面的地理坐标投影到平面上的一种方法。GeoTools提供了许多常用的投影方法和投影坐标系统的支持。
4. **数据格式（Data Formats）：** GeoTools支持多种地理空间数据格式，如Shapefile、GeoJSON、KML、GML等，可以方便地读取和写入这些数据格式。
5. **空间分析（Spatial Analysis）：** GeoTools提供了丰富的空间分析功能，包括缓冲区分析、空间查询、空间叠加分析等，可以帮助用户进行地理空间

数据的处理和分析。

总的来说，GeoTools是一个功能丰富的GIS工具库，提供了丰富的功能和工具，可以帮助用户处理和分析各种地理空间数据，并构建地理空间应用。

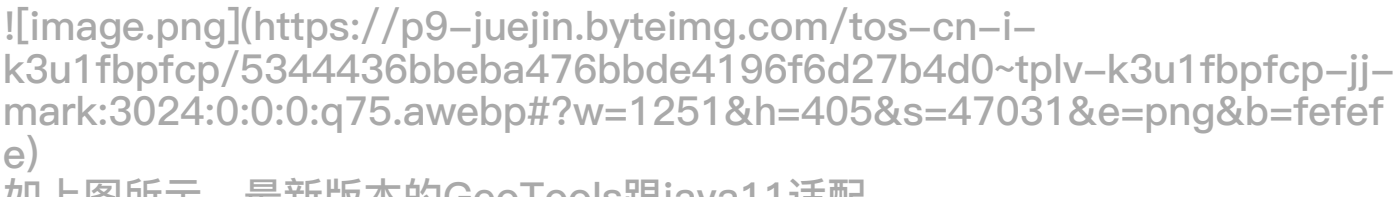
3. Geotools使用示例

3.1. 开发环境搭建

3.1.1. 所需版本和工具

依赖	版本
Spring Boot	2.6.14
GeoTools	27-SNAPSHOT
java	1.8以上
ArcGis	10.8

我这里用的不是GeoTools的最新版本，需要最新版本的同学可登录GeoTools的官网（[GeoTools The Open Source Java GIS Toolkit — GeoTools](http://cxyroad.com/ "https://geotools.org/")）查看最新版本和其使用规则。



如上图所示，最新版本的GeoTools跟java11适配。

3.1.2. pom依赖

```
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-shapefile</artifactId>
            <version>${geotools.version}</version>
         </dependency>
```

```
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-geojson</artifactId>
            <version>${geotools.version}</version>
        </dependency>
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-swing</artifactId>
            <version>${geotools.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
```

4. 工具类介绍

4.1. 读取shp工具类

4.1.1 准备数据

打开ArcGis绘制面数据，我这里绘制了4490坐标系的几个面要素。

4.1.2. 部分方法

1. 读取shp中的空间要素信息（wkt）

```
public static List<String> getWktFromShp(String shpPath) {
        List<String> shpList = new ArrayList<>();
        SimpleFeatureCollection simpleFeatureCollection = null;
        try {
            //获取文件
            File file = new File(shpPath);
            // 读取到数据存储中
            FileDataStore dataStore =
FileDataStoreFinder.getDataStore(file);
            // 获取特征资源
            SimpleFeatureSource simpleFeatureSource =
dataStore.getFeatureSource();
            // 要素集合
            simpleFeatureCollection = simpleFeatureSource.getFeatures();
        } catch (IOException e) {
            e.printStackTrace();
        }
        // 获取要素迭代器
        SimpleFeatureIterator featureIterator =
simpleFeatureCollection.features();
        while (featureIterator.hasNext()) {
            // 要素对象
            SimpleFeature feature = featureIterator.next();
            Object geometryText = feature.getDefaultGeometry();
            log.info(geometryText.toString());
            shpList.add(geometryText.toString());

        }
        featureIterator.close();
        return shpList;
    }
```

2. 运行结果

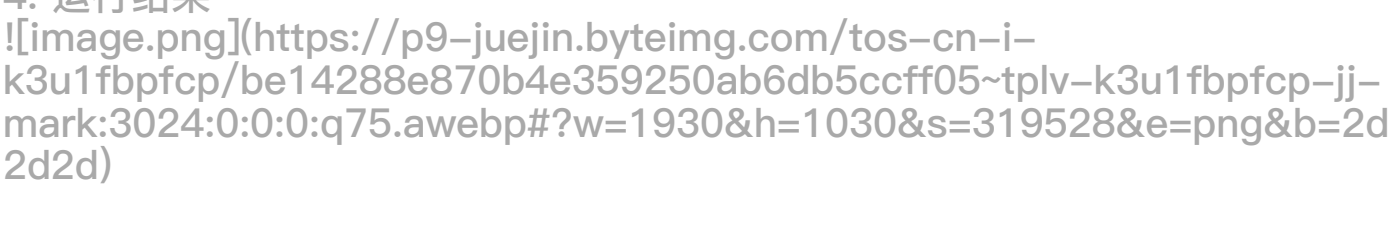
3. 读取shp文件并将其转换为Geojson

```
/**
```

```
     * 构造Geojson结构体
     * @param featuresJson
     * @return
     */
    public static JSONObject buildGeoJson(JSONArray featuresJson) {
        JSONObject Geojson = new JSONObject();
        Geojson.put("type", "FeatureCollection");
        Geojson.put("features", featuresJson);
        return Geojson;
    }
 /**
     * 构造Geojson的features部分 单个
     *
     * @param geoObject
     * @param properties
     * @return
     */
    public static JSONObject buildFeature(Map geoObject, Map
properties) {
        JSONObject featureObject = new JSONObject();
        Map featureMap = new HashMap();
        featureMap.put("type", "Feature");
        featureMap.putAll(geoObject);
        featureMap.put("properties", properties);
        featureObject.putAll(featureMap);
        return featureObject;
    }
 /**
     * 获取空间信息并构造为Map
     * @param wkt
     * @return
     */
    public static Map getGeoMap(String wkt) {
        Map<String, Object> geoMap = new HashMap<>();
        String json = null;
        try {
            WKTReader reader = new WKTReader();
            Geometry geometry = reader.read(wkt);
            StringWriter writer = new StringWriter();
            GeometryJSON g = new GeometryJSON();
            g.write(geometry, writer);
            geoMap.put("geometry", writer);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return geoMap;
    }
```

```
 /**
     * 基于shp构造geojson并返回
     *
     * @param shpPath
     * @return
     */
    public static JSONObject buildGeojsonFromShp(String shpPath) {
        JSONArray featureArray = new JSONArray();
//        List<String>shpList=new ArrayList<>();
        SimpleFeatureCollection simpleFeatureCollection = null;
        try {
//            要素合集
            //获取文件
            File file = new File(shpPath);
            // 读取到数据存储中
            ShapefileDataStore dataStore = (ShapefileDataStore)
FileDataStoreFinder.getDataStore(file);
            dataStore.setCharset(Charset.forName("GBK"));
            // 获取特征资源
            SimpleFeatureSource simpleFeatureSource =
dataStore.getFeatureSource();
            // 要素集合
            simpleFeatureCollection = simpleFeatureSource.getFeatures();
        } catch (IOException e) {
            e.printStackTrace();
        }
        SimpleFeatureIterator featureIterator =
simpleFeatureCollection.features();
//            // 要素数量
        int featureSize = simpleFeatureCollection.size();
//            log.info("要素数量"+featureSize);
        //创建properties  Map
        while (featureIterator.hasNext()) {
            // 要素对象
            SimpleFeature feature = featureIterator.next();
            Collection<Property> propertyCollection =
(Collection<Property>) feature.getValue();
            //填充属性map
            Map<String, Object> properMap = new HashMap<>();
            for (Property property : propertyCollection) {
                if (property.getName().toString().equals("the_geom")) {
                    continue;
                }
                properMap.put(property.getName().toString(),
property.getValue());
            }
            //获取geo信息
            Object geometryText = feature.getDefaultGeometry();
```

```
            Map geoMap = getGeoMap(geometryText.toString());
            JSONObject featureObject = buildFeature(geoMap, properMap);
            featureArray.add(featureObject);
        }
        featureIterator.close();
        JSONObject GeoJson = buildGeoJson(featureArray);

        return GeoJson;
    }


```

4. 运行结果


4.1.3. 完整工具类代码

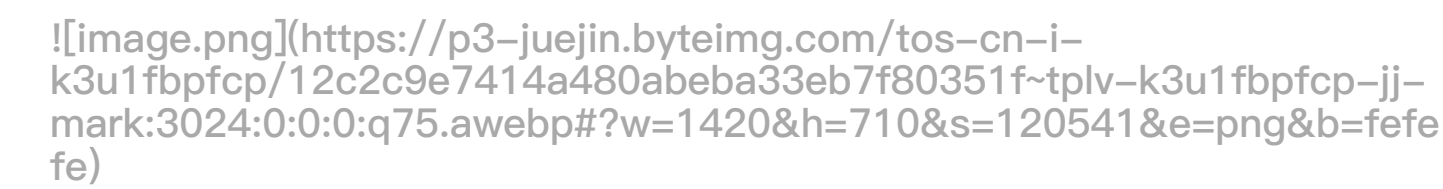
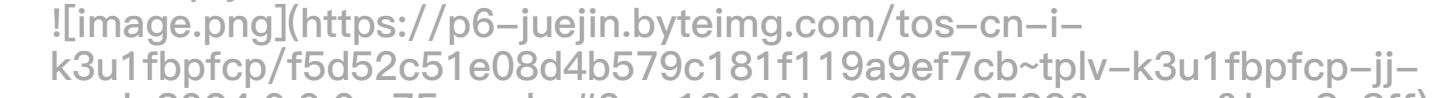
```
import lombok.extern.slf4j.Slf4j;
import org.geotools.data.FileDataStore;
import org.geotools.data.FileDataStoreFinder;
import org.geotools.data.shapefile.ShapefileDataStore;
import org.geotools.data.simple.SimpleFeatureCollection;
import org.geotools.data.simple.SimpleFeatureIterator;
import org.geotools.data.simple.SimpleFeatureSource;
import org.geotools.geojson.geom.GeometryJSON;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.locationtech.jts.geom.Geometry;
import org.locationtech.jts.geom.GeometryFactory;
import org.locationtech.jts.geom.Point;
import org.locationtech.jts.io.ParseException;
import org.locationtech.jts.io.WKTReader;
import org.locationtech.jts.io.WKTWriter;
import org.opengis.feature.Property;
import org.opengis.feature.simple.SimpleFeature;
import java.io.File;
import java.io.IOException;
import java.io.StringWriter;
import java.nio.charset.Charset;
import java.util.*;
```

```
/**
 * @version 1.0
 * @description: gis工具类
 * @author: chenss
 * @date 2024-03-14 16:19
 */
@Slf4j
public class GisUtil {
    public static void main(String[] args) {
        JSONObject
Geojson=buildGeojsonFromShp("D:\\arcgisdata\\mesh4490.shp");
        log.info(Geojson.toJSONString());
//        List<String> wkts =
getWktFromShp("D:\\arcgisdata\\mesh4490.shp");
//        for (String wkt : wkts) {
//            log.info(wkt);
//        }
    }

    /**
     * 构造Geojson结构体
     * @param featuresJson
     * @return
     */
    public static JSONObject buildGeoJson(JSONArray featuresJson) {
        JSONObject Geojson = new JSONObject();
        Geojson.put("type", "FeatureCollection");
        Geojson.put("features", featuresJson);
        return Geojson;
    }

    /**
     * 构造Geojson的features部分 单个
     *
     * @param geoObject
     * @param properties
     * @return
     */
    public static JSONObject buildFeature(Map geoObject, Map
properties) {
        JSONObject featureObject = new JSONObject();
        Map featureMap = new HashMap();
        featureMap.put("type", "Feature");
        featureMap.putAll(geoObject);
        featureMap.put("properties", properties);
        featureObject.putAll(featureMap);
        return featureObject;
```

```
    }

    /**
     * 获取空间信息并构造为Map
     * @param wkt
     * @return
     */
    public static Map getGeoMap(String wkt) {
        Map<String, Object> geoMap = new HashMap<>();
        String json = null;
        try {
            WKTReader reader = new WKTReader();
            Geometry geometry = reader.read(wkt);
            StringWriter writer = new StringWriter();
            GeometryJSON g = new GeometryJSON();
            g.write(geometry, writer);
            geoMap.put(”geometry”, writer);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return geoMap;
    }

    /**
     * 只读取geo信息 wkt
     *
     * @param shpPath
     * @return
     */
    public static List<String> getWktFromShp(String shpPath) {
        List<String> shpList = new ArrayList<>();
        SimpleFeatureCollection simpleFeatureCollection = null;
        try {
            //获取文件
            File file = new File(shpPath);
            // 读取到数据存储中
            FileDataStore dataStore =
FileDataStoreFinder.getDataStore(file);
            // 获取特征资源
            SimpleFeatureSource simpleFeatureSource =
dataStore.getFeatureSource();
            // 要素集合
            simpleFeatureCollection = simpleFeatureSource.getFeatures();
        } catch (IOException e) {
            e.printStackTrace();
        }
        // 获取要素迭代器
        SimpleFeatureIterator featureIterator =
```

```
simpleFeatureCollection.features();
        while (featureIterator.hasNext()) {
            // 要素对象
            SimpleFeature feature = featureIterator.next();
            Object geometryText = feature.getDefaultGeometry();
            log.info(geometryText.toString());
            shpList.add(geometryText.toString());

        }
        featureIterator.close();
        return shpList;
    }

    /**
     * 基于shp构造geojson并返回
     *
     * @param shpPath
     * @return
     */
    public static JSONObject buildGeojsonFromShp(String shpPath) {
        JSONArray featureArray = new JSONArray();
//        List<String>shpList=new ArrayList<>();
        SimpleFeatureCollection simpleFeatureCollection = null;
        try {
//            要素合集
            //获取文件
            File file = new File(shpPath);
            // 读取到数据存储中
            ShapefileDataStore dataStore = (ShapefileDataStore)
FileDataStoreFinder.getDataStore(file);
            dataStore.setCharset(Charset.forName("GBK"));
            // 获取特征资源
            SimpleFeatureSource simpleFeatureSource =
dataStore.getFeatureSource();
            // 要素集合
            simpleFeatureCollection = simpleFeatureSource.getFeatures();
        } catch (IOException e) {
            e.printStackTrace();
        }
        SimpleFeatureIterator featureIterator =
simpleFeatureCollection.features();
//            // 要素数量
        int featureSize = simpleFeatureCollection.size();
//            log.info("要素数量"+featureSize);
        //创建properties  Map
        while (featureIterator.hasNext()) {
            // 要素对象
            SimpleFeature feature = featureIterator.next();
```

```
            Collection<Property> propertyCollection =
(Collection<Property>) feature.getValue();
            //填充属性map
            Map<String, Object> properMap = new HashMap<>();
            for (Property property : propertyCollection) {
                if (property.getName().toString().equals("the_geom")) {
                    continue;
                }
                properMap.put(property.getName().toString(),
property.getValue());
            }
            //获取geo信息
            Object geometryText = feature.getDefaultGeometry();
            Map geoMap = getGeoMap(geometryText.toString());
            JSONObject featureObject = buildFeature(geoMap, properMap);
            featureArray.add(featureObject);
        }
        featureIterator.close();
        JSONObject GeoJson = buildGeoJson(featureArray);

        return GeoJson;
    }

    /**
     * 根据给定的wkt面求出中心点，并以wkt形式返回
     */
    public static String calculateCenter(String wktPolygon) throws
ParseException {
        // 创建 WKT 解析器和写入器
        WKTReader reader = new WKTReader(new GeometryFactory());
        WKTWriter writer = new WKTWriter();

        // 解析面的几何对象
        Geometry geometry = reader.read(wktPolygon);

        // 计算面的中心点
        Point center = geometry.getCentroid();

        // 将中心点转换为 WKT 格式
        String wktCenter = writer.write(center);

        return wktCenter;
    }


}
```

```

### 4.2. 坐标转换工具类

我这个坐标转换工具只应用于同椭球（本文示例为2000坐标系-EPSG：4490）坐标投影转换。已云南的投影带为33-35,对应的EPSG为4521、4522、4523。

#### 4.2.1. 准备数据

1. 准备4490、4521、4522、4523的shp




2. 获取.prj中坐标描述信息




3. 将坐标描述信息存放到常量Map中

```
static final Map<String,String> projMap=new HashMap();
static {
projMap.put("4522","PROJCS["CGCS2000_3_Degree_GK_Zone_34",GEOGCS["GCS_China_Geodetic_Coordinate_System_2000",DATUM["D_China_2000",SPHEROID["CGCS2000",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Gauss_Kruger"],PARAMETER["False_Easting",34500000.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",102.0],PARAMETER["Scale_Factor",1.0],PARAMETER["Latitude_Of_Origin",0.0],UNIT["Meter",1.0]]");
    projMap.put("4490","GEOGCS["China Geodetic Coordinate System 2000",\n" +
            "   DATUM["China_2000",\n" +
            "   SPHEROID["CGCS2000",6378137,298.257222101,\n" +
            "   AUTHORITY["EPSG","1024"]],\n" +
            "   AUTHORITY["EPSG","1043"]],\n" +
            "   PRIMEM["Greenwich",0,\n" +
            "   AUTHORITY["EPSG","8901"]],\n" +
            "   UNIT["degree",0.0174532925199433,\n" +
            "   AUTHORITY["EPSG","9122"]],\n" +
            "   AUTHORITY["EPSG","4490"]]");
```
```

projMap.put(”4521”,”PROJCS[”CGCS2000_3_Degree_GK_Zone_33”,GEOGCS[”GCS_China_Geodetic_Coordinate_System_2000”,DATUM[”D_China_2000”,SPHEROID[”CGCS2000”,6378137.0,298.257222101]],PRIMEM[”Greenwich”,0.0],UNIT[”Degree”,0.0174532925199433]],PROJECTION[”Gauss_Kruger”],PARAMETER[”False_Easting”,33500000.0],PARAMETER[”False_Northing”,0.0],PARAMETER[”Central_Meridian”,99.0],PARAMETER[”Scale_Factor”,1.0],PARAMETER[”Latitude_Of_Origin”,0.0],UNIT[”Meter”,1.0]]”);
projMap.put(”4523”,”PROJCS[”CGCS2000_3_Degree_GK_Zone_35”,GEOGCS[”GCS_China_Geodetic_Coordinate_System_2000”,DATUM[”D_China_2000”,SPHEROID[”CGCS2000”,6378137.0,298.257222101]],PRIMEM[”Greenwich”,0.0],UNIT[”Degree”,0.0174532925199433]],PROJECTION[”Gauss_Kruger”],PARAMETER[”False_Easting”,35500000.0],PARAMETER[”False_Northing”,0.0],PARAMETER[”Central_Meridian”,105.0],PARAMETER[”Scale_Factor”,1.0],PARAMETER[”Latitude_Of_Origin”,0.0],UNIT[”Meter”,1.0]]”);
}

```

#### 4.2.2. 完整工具类代码

```
import lombok.extern.slf4j.Slf4j;
import org.geotools.data.FeatureWriter;
import org.geotools.data.FileDataStoreFactorySpi;
import org.geotools.data.Transaction;
import org.geotools.data.shapefile.ShapefileDataStore;
import org.geotools.data.shapefile.ShapefileDataStoreFactory;
import org.geotools.data.simple.SimpleFeatureIterator;
import org.geotools.data.simple.SimpleFeatureSource;
import org.geotools.feature.simple.SimpleFeatureTypeBuilder;
import org.geotools.geometry.jts.JTS;
import org.geotools.referencing.CRS;
import org.locationtech.jts.geom.Geometry;
import org.opengis.feature.simple.SimpleFeature;
import org.opengis.feature.simple.SimpleFeatureType;
import org.opengis.referencing.crs.CoordinateReferenceSystem;
import org.opengis.referencing.operation.MathTransform;
import java.io.File;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

@Slf4j
```
```

```
public class ProjTransUtil {
    static final  Map<String,String> projMap=new HashMap();
    static {
projMap.put("4522","PROJCS[\"CGCS2000_3_Degree_GK_Zone_34\",GEOGCS[\"GCS_China_Geodetic_Coordinate_System_2000\",DATUM[\"D_China_2000\",SPHEROID[\"CGCS2000\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]],PROJECTION[\"Gauss_Kruger\"],PARAMETER[\"False_Easting\",34500000.0],PARAMETER[\"False_Northing\",0.0],PARAMETER[\"Central_Meridian\",102.0],PARAMETER[\"Scale_Factor\",1.0],PARAMETER[\"Latitude_Of_Origin\",0.0],UNIT[\"Meter\",1.0]]");
        projMap.put("4490","GEOGCS[\"China Geodetic Coordinate System 2000\",\n" +
                "    DATUM[\"China_2000\",\n" +
                "
SPHEROID[\"CGCS2000\",6378137,298.257222101,\n" +
                "            AUTHORITY[\"EPSG\",\"1024\"]],\n" +
                "        AUTHORITY[\"EPSG\",\"1043\"]],\n" +
                "    PRIMEM[\"Greenwich\",0,\n" +
                "        AUTHORITY[\"EPSG\",\"8901\"]],\n" +
                "    UNIT[\"degree\",0.0174532925199433,\n" +
                "        AUTHORITY[\"EPSG\",\"9122\"]],\n" +
                "    AUTHORITY[\"EPSG\",\"4490\"]]");
projMap.put("4521","PROJCS[\"CGCS2000_3_Degree_GK_Zone_33\",GEOGCS[\"GCS_China_Geodetic_Coordinate_System_2000\",DATUM[\"D_China_2000\",SPHEROID[\"CGCS2000\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]],PROJECTION[\"Gauss_Kruger\"],PARAMETER[\"False_Easting\",33500000.0],PARAMETER[\"False_Northing\",0.0],PARAMETER[\"Central_Meridian\",99.0],PARAMETER[\"Scale_Factor\",1.0],PARAMETER[\"Latitude_Of_Origin\",0.0],UNIT[\"Meter\",1.0]]");
projMap.put("4523","PROJCS[\"CGCS2000_3_Degree_GK_Zone_35\",GEOGCS[\"GCS_China_Geodetic_Coordinate_System_2000\",DATUM[\"D_China_2000\",SPHEROID[\"CGCS2000\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]],PROJECTION[\"Gauss_Kruger\"],PARAMETER[\"False_Easting\",35500000.0],PARAMETER[\"False_Northing\",0.0],PARAMETER[\"Central_Meridian\",105.0],PARAMETER[\"Scale_Factor\",1.0],PARAMETER[\"Latitude_Of_Origin\",0.0],UNIT[\"Meter\",1.0]]");
    }

    /**
     * 根据传入wkt获取数据坐标系
     * @param wkt
     * @return
     */
    public static String getProj(String wkt){
        String resEpsg="";
```

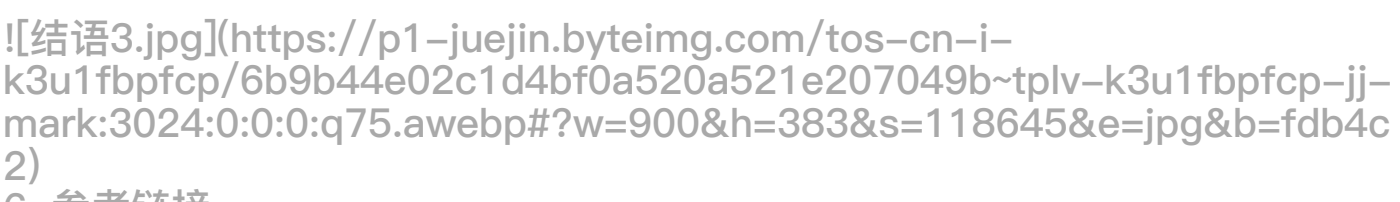
```
        int lonLatStart=-1;
        //根据wkt字符串判断平面坐标位于哪一度带
        try {
            for (int i = 0; i < wkt.length(); i++) {
if(Integer.valueOf(wkt.charAt(i))>=48&&Integer.valueOf(wkt.charAt(i))<58)
{
                    lonLatStart=i;
                    break;
                }
            }
            int lonLatEnd=wkt.indexOf(",");
            String projLonLat=wkt.substring(lonLatStart,lonLatEnd);
            String[]lonlat=projLonLat.split(" ");
            String projLon=lonlat[0];
            if(projLon.substring(0,2).equals("33")){
                resEpsg="4521";
            }else if(projLon.substring(0,2).equals("34")){
                resEpsg="4522";
            }else if(projLon.substring(0,2).equals("35")){
                resEpsg="4523";
            }
            else
                return "4490";
        } catch (Exception e) {
            log.info(wkt+"出错");
            log.error(e.getMessage(),e);
        }
        return resEpsg;

    }

    /**
     * 坐标转换
     * @param geom
     * @param sourceEpsg
     * @return
     */
    public static Geometry lonlat2WebMactor(Geometry geom,String
sourceEpsg){
        try{
            //这里是以OGC WKT形式定义的是World Mercator投影，网页地图
一般使用该投影
            CoordinateReferenceSystem
crsSource=CRS.parseWKT(projMap.get(sourceEpsg));
            CoordinateReferenceSystem crsTarget =
CRS.parseWKT(projMap.get("4490"));
            // 投影转换
//            MathTransform transform =
```

```
CRS.findMathTransform(DefaultGeographicCRS.WGS84, crsTarget);
            MathTransform transform = CRS.findMathTransform(crsSource,
crsTarget);
            return JTS.transform(geom, transform);
        }
        catch (Exception e) {
            // TODO Auto-generated catch block
            log.error(e.getMessage(),e);
            return null;
        }
    }

    /**
     *给定inputshp转换为targetEpsg坐标系，并输出到outputShp位置
     * @param inputShp
     * @param outputShp
     * @param targetEpsg
     * @return
     */
    public static Map projectShape(String inputShp, String
outputShp,String targetEpsg){
        Map map = new HashMap();
        try {
            //源shape文件
            ShapefileDataStore shapeDS = (ShapefileDataStore) new
ShapefileDataStoreFactory().createDataStore(new
File(inputShp).toURI().toURL());
            //创建目标shape文件对象
            Map<String, Serializable> params = new HashMap<String,
Serializable>();
            FileDataStoreFactorySpi factory = new
ShapefileDataStoreFactory();
            File file=FileUtil.createFileByPath(outputShp);
//            if(!file.exists()){
params.put(ShapefileDataStoreFactory.URLP.key,file.toURI().toURL());
           ShapefileDataStore ds = (ShapefileDataStore)
factory.createNewDataStore(params);
//            Charset charset = Charset.forName("UTF-8");
//            ds.setCharset(charset);
             // 设置属性
             SimpleFeatureSource fs =
shapeDS.getFeatureSource(shapeDS.getTypeNames()[0]);
              //下面这行还有其他写法，根据源shape文件的
simpleFeatureType可以不用retype，而直接用fs.getSchema设置
//            CoordinateReferenceSystem crs =
CRS.parseWKT(strWKTMercator);
              CoordinateReferenceSystem crs =
CRS.parseWKT(projMap.get("4490"));
```

```
ds.createSchema(SimpleFeatureTypeBuilder.retype(fs.getSchema(), crs));


                //设置writer
                FeatureWriter<SimpleFeatureType, SimpleFeature> writer =
ds.getFeatureWriter(ds.getTypeNames()[0], Transaction.AUTO_COMMIT);

                //写记录
                SimpleFeatureIterator it = fs.getFeatures().features();
                try {
                    while (it.hasNext()) {
                        SimpleFeature f = it.next();
                        SimpleFeature fNew = writer.next();
                        fNew.setAttributes(f.getAttributes());
                        Geometry geom =
lonlat2WebMactor((Geometry)f.getAttribute("the_geom"),targetEpsg);
                        fNew.setAttribute("the_geom", geom);
                    }
                }
                finally {
                    it.close();
                }
                writer.write();
                writer.close();
                ds.dispose();
                shapeDS.dispose();
//              }
            map.put("status", "success");
            map.put("message", outputShp);
        }
        catch (Exception e) {
            log.error(e.getMessage(),e);
            map.put("status", "failure");
            map.put("message", e.getMessage());
        }
        return map;
    }

//    public static void main(String[] args) {
//        String input="D:\\jsonshp\\test.shp";
//        String output="D:\\jsonshp\\test4490.shp";
//        projectShape(input,output);
//    }

}


```

5.结语

本文对Geotools的基本概念进行了简介，之后介绍了Geotools的工具类及其具体用法。下一篇文章将讲解Postgis+Geotools+MybatisPlus实现数据的读取，写入及前端展示。对Gis开发领域感兴趣的同学可动动你们发财的小手点点~



6. 参考链接

[常见2000坐标系对应的EPSG代号 - 知乎](http://cxyroad.com/”https://zhuanlan.zhihu.com/p/488194364?utm_id=0”)

原文链接: https://juejin.cn/post/7346837205774696482