智慧城市

文档版本记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 作者 | 日期 | 版本 | 纪要 |
|  | 李坡 | 2019年7月20日 | V1.0.0 | 文档内容形式 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[目录 3](#_Toc29895547)

[1 编写目的 4](#_Toc29895548)

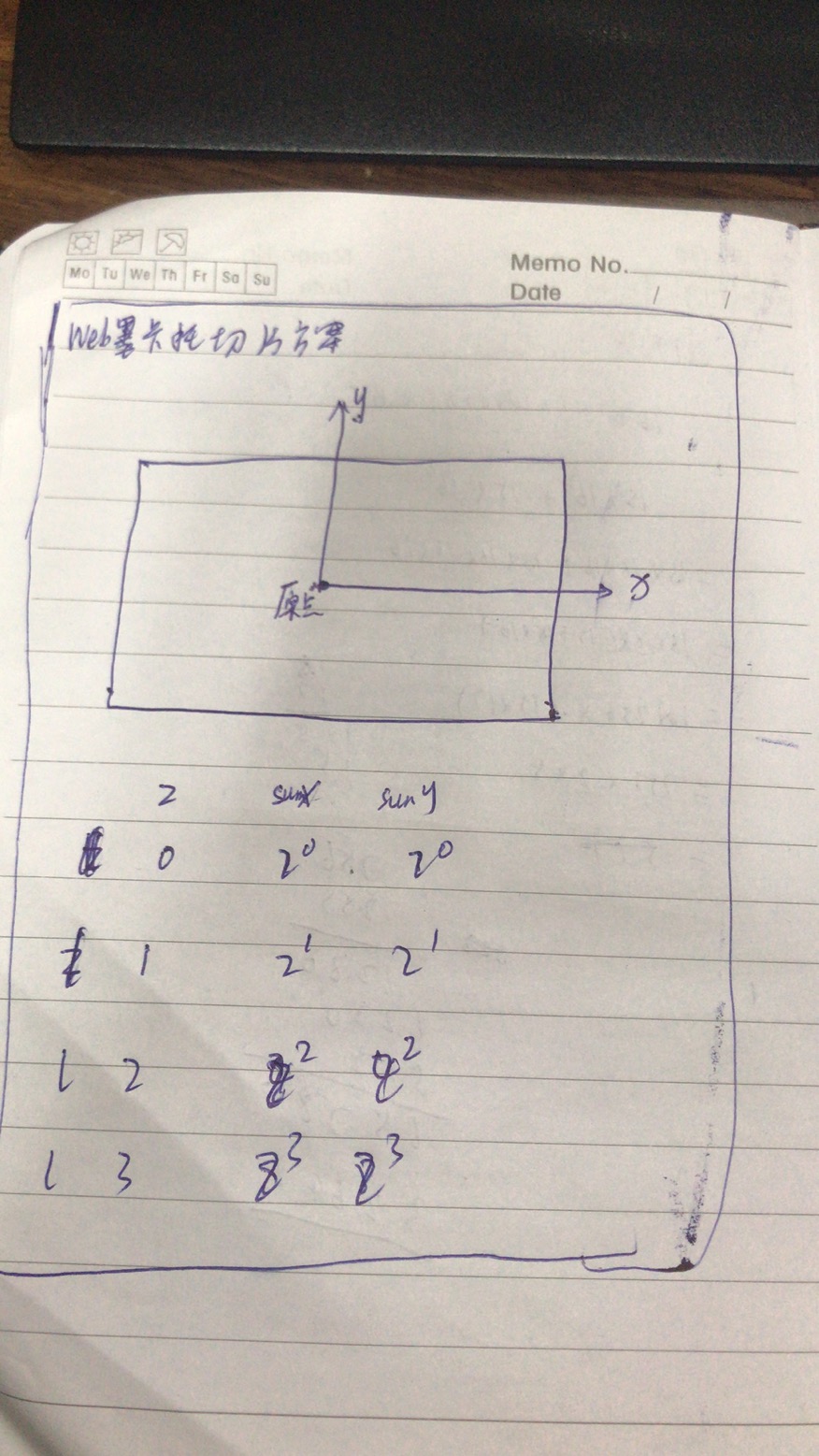
[2 Web墨卡托切片方案 4](#_Toc29895549)

[3 地理切片方案 9](#_Toc29895550)

# 编写目的

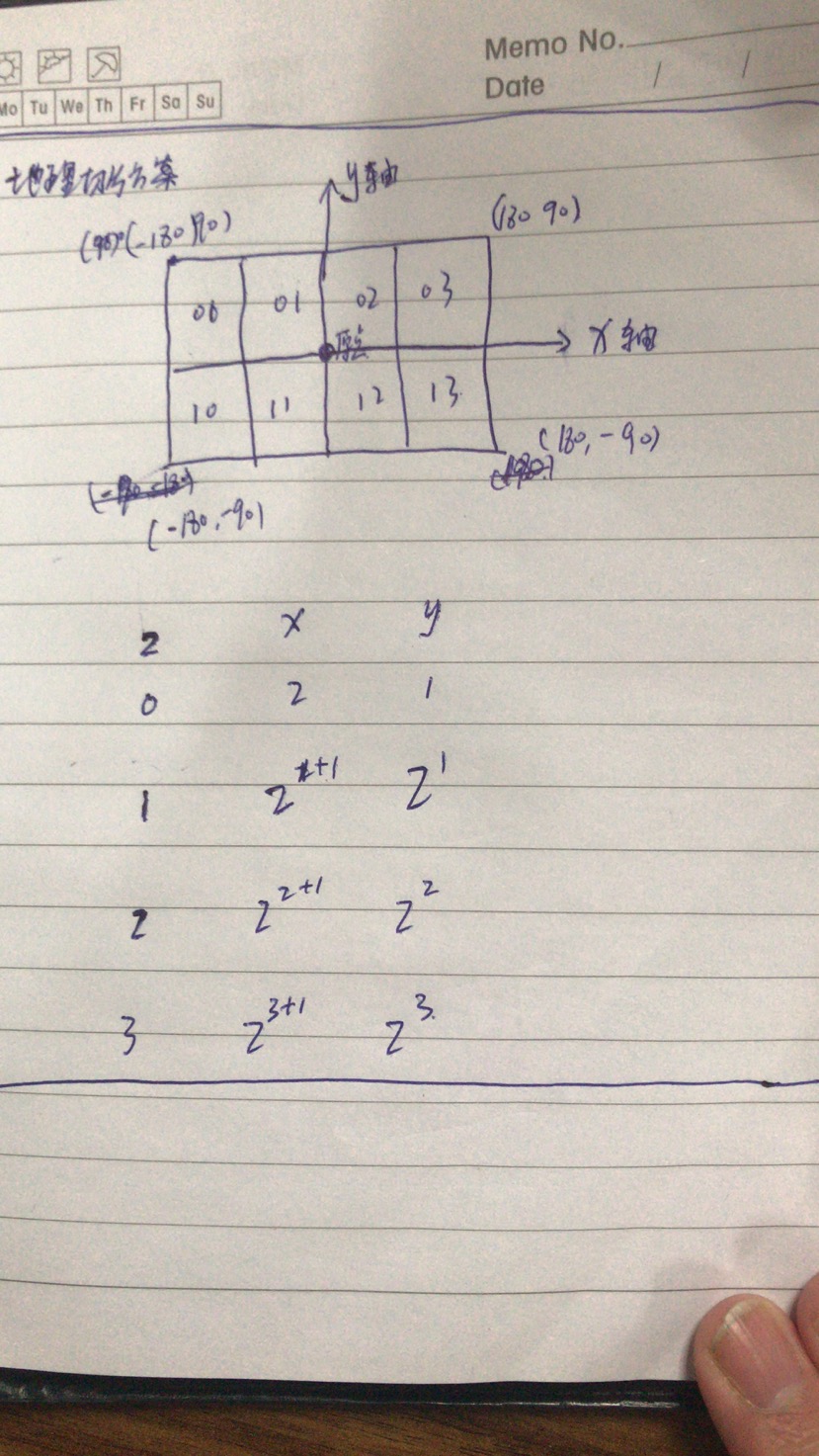
该文档主要是描述Web墨卡托切片方案和地理切片方案。

# Web墨卡托切片方案



|  |
| --- |
| public static double *minx* = -20037508.3427892; public static double *maxx* = 20037508.3427892;  */\*\*  \* 切片转范围  \*   \** ***@param*** *x x轴切片的编号  \** ***@param*** *y y轴方向切片的编号  \** ***@param*** *level 切片的级别  \** ***@return*** *\*/* public static Envelope tileXYToNativeRectangle(int x, int y, int level) {  //根据web墨卡托切片方案，得到该级别下x轴的切片总个数和y轴的切片总个数  int xTiles = *getNumberOfXTilesAtLevel*(level);  int yTiles = *getNumberOfYTilesAtLevel*(level);   //获取x轴，该级别下每个切片的宽度  double xTileWidth = (*maxx* - *minx*) / xTiles;  //根据切片的索引编号，获取该切片的x轴的最小最大值  double west = *minx* + x \* xTileWidth;  double east = *minx* + (x + 1) \* xTileWidth;   //获取y轴，该级别下每个切片的宽度  double yTileHeight = (*maxx* - *minx*) / yTiles;  double north = *maxx* - y \* yTileHeight;  double south = *maxx* - (y + 1) \* yTileHeight;  Envelope envelope = new Envelope(west, east, north, south);   return envelope; }  public static Map<String, Integer> positionToTileXY(int level, double longitude, double latitude) {  Map<String, Integer> map = new HashMap<String, Integer>();   */\*\*  \* 经纬度转为墨卡托  \*/* Coordinate coordinate = new Coordinate(longitude, latitude);  Coordinate c = *lonLat2Mercator*(coordinate);   // 计算在该级别下X方向切片数量  int xTiles = *getNumberOfXTilesAtLevel*(level);  // 计算在该级别下Y方向切片数量  int yTiles = *getNumberOfYTilesAtLevel*(level);   double overallWidth = *maxx* - *minx*;  double xTileWidth = overallWidth / xTiles;  double overallHeight = *maxx* - *minx*;  double yTileHeight = overallHeight / yTiles;   double distanceFromWest = c.x - *minx*;  double distanceFromNorth = *maxx* - c.y;   double xTileCoordinate = distanceFromWest / xTileWidth;  if (xTileCoordinate >= xTiles) {  xTileCoordinate = xTiles - 1;  }  map.put("X", (int) xTileCoordinate);   double yTileCoordinate = distanceFromNorth / yTileHeight;  if (yTileCoordinate >= yTiles) {  yTileCoordinate = yTiles - 1;  }  map.put("Y", (int) yTileCoordinate);   return map; }    */\*\*  \* 获得某个范围下某个级别下的切片XY的范围 ,比如X:[10,20], Y:[20,30]  \*  \** ***@param*** *level  \* -- 缩放级别  \** ***@param*** *minX  \* -- 最小X  \** ***@param*** *maxX  \* -- 最大X  \** ***@param*** *minY  \* -- 最小Y  \** ***@param*** *maxY  \* -- 最大Y  \** ***@return*** *\*/* public static Map<String, Integer> getTileXYRange(int level, double minX, double maxX, double minY,  double maxY) {   Map<String, Integer> minXY = *positionToTileXY*(level, minX, minY);  Map<String, Integer> maxXY = *positionToTileXY*(level, maxX, maxY);   Integer minTileX = minXY.get("X");  Integer maxTileX = maxXY.get("X");  Integer minTileY = minXY.get("Y");  Integer maxTileY = maxXY.get("Y");   Map<String, Integer> mapResult = new HashMap<>();  mapResult.put("minTileX", minTileX);  mapResult.put("maxTileX", maxTileX);  mapResult.put("minTileY", minTileY);  mapResult.put("maxTileY", maxTileY);   return mapResult;  }    private static Integer getNumberOfXTilesAtLevel(int level) {  return 1 << level; }  private static Integer getNumberOfYTilesAtLevel(int level) {  return 1 << level; }  */\*\*  \* 经纬度转墨卡托  \*   \** ***@param*** *lonLat  \** ***@return*** *\*/* public static Coordinate lonLat2Mercator(Coordinate lonLat) {   double x = lonLat.x \* 20037508.34 / 180;  double y = Math.*log*(Math.*tan*((90 + lonLat.y) \* Math.*PI* / 360)) / (Math.*PI* / 180);  y = y \* *maxx* / 180;   return new Coordinate(x, y); }  // 墨卡托转经纬度 public static Coordinate mercator2LonLat(Coordinate mercator) {   double x = mercator.x / 20037508.34 \* 180;  double y = mercator.y / 20037508.34 \* 180;  y = 180 / Math.*PI* \* (2 \* Math.*atan*(Math.*exp*(y \* Math.*PI* / 180)) - Math.*PI* / 2);   return new Coordinate(x, y); } |

# 地理切片方案



|  |
| --- |
| package onegis.geoservice.map;  import com.vividsolutions.jts.geom.Coordinate; import com.vividsolutions.jts.geom.Envelope; import com.vividsolutions.jts.geom.GeometryFactory; import com.vividsolutions.jts.geom.Polygon;  import java.util.HashMap; import java.util.Map;  public class GeographicTilingScheme {   private Integer getNumberOfXTilesAtLevel(int level) {  return 2 << level;  }   private Integer getNumberOfYTilesAtLevel(int level) {  return 1 << level;  }   */\*\*  \* 切片转矩形  \*  \** ***@param*** *x  \** ***@param*** *y  \** ***@param*** *level  \** ***@return*** *\*/* public static Envelope tileXYToRectangle(int x, int y, int level) {  //根据地理坐标切片方案，得到该级别下x轴的切片总个数和y轴的切片总个数  GeographicTilingScheme g = new GeographicTilingScheme();  int xTiles = g.getNumberOfXTilesAtLevel(level);  int yTiles = g.getNumberOfYTilesAtLevel(level);   //获取x轴，该级别下每个切片的宽度  double xTileWidth = 360d / xTiles;  double west = x \* xTileWidth + (-180);  double east = (x + 1) \* xTileWidth + (-180);   double yTileHeight = 180d / yTiles;  double north = (90) - (y) \* yTileHeight;   double south = (90) - (y + 1) \* yTileHeight;   Envelope envelope = new Envelope(west, east, north, south);  return envelope;  }   */\*\*  \* 经纬度转切片  \*  \** ***@param*** *level  \** ***@param*** *longitude  \** ***@param*** *latitude  \** ***@return*** *\*/* public static Map<String, Integer> positionToTileXY(int level, double longitude, double latitude) {  GeographicTilingScheme g = new GeographicTilingScheme();  Map<String, Integer> map = new HashMap<String, Integer>();   int xTiles = g.getNumberOfXTilesAtLevel(level);  int yTiles = g.getNumberOfYTilesAtLevel(level);   double xTileWidth = 360d / xTiles;  double yTileHeight = 180d / yTiles;   double xTileCoordinate = (longitude - (-180)) / xTileWidth;  if (xTileCoordinate >= xTiles) {  xTileCoordinate = xTiles - 1;  }  map.put("X", (int) xTileCoordinate);  double yTileCoordinate = (90 - latitude) / yTileHeight;  if (yTileCoordinate >= yTiles) {  yTileCoordinate = yTiles - 1;  }  map.put("Y", (int) yTileCoordinate);   return map;   }   */\*\*  \* 根据切片获取范围  \*  \** ***@param*** *x  \** ***@param*** *y  \** ***@param*** *level  \** ***@return*** *\*/* public static Polygon tileXYToGeometry(int x, int y, int level) {  Envelope r = GeographicTilingScheme.*tileXYToRectangle*(x, y, level);   GeometryFactory geometryFactory = new GeometryFactory();  Coordinate[] coords = new Coordinate[] { new Coordinate(r.getMinX(), r.getMinY()),  new Coordinate(r.getMaxX(), r.getMaxY()), new Coordinate(r.getMinX(), r.getMaxY()),  new Coordinate(r.getMaxX(), r.getMinY()) };   Polygon polygon = geometryFactory.createPolygon(coords);  return polygon;  }   public static void main(String[] args) throws Exception {   // GeographicTilingScheme g = new GeographicTilingScheme();  // System.out.println(g.positionToTileXY(15, 113.42, 34.5));   }  } |