tiletanic Documentation

Release 0.0.5

Patrick Young

May 18, 2021

Contents

Tiletanic is a library for making and using geospatial tiling schemes. It's goal is to provide tooling for dealing with the conversion between a tile specified in as (row, column, zoom level), geospatial coordinates, or quadkeys. It also provides functionaly for taking an input geometry and figuring out what tiles cover it.

Tiletanic is MIT licensed.

Contributions are more than welcome! Please check out the GitHub site.

Installation is easy:

```
pip install tiletanic
```

The only dependency is shapely, and to install that, you'll need GEOS installed (usually in your package manager).

CHAPTER 1

Motivation

You may be familar with the Web Mercator (or Spherical Mercator, Google Tiles, etc) tiling scheme, which is the most commonly encountered tiling scheme on the web. There are good tools out there for dealing specifically with this projection, for instance Mercantile fits the bill. If you're dealing exclusively with this projection, you might get better mileage with them!

One oddity of Web Mercator is that coordinates are usually expressed in geographic (longitude, latitude) coordinates that live on the ellipsoid (WGS84) rather than in units of the Web Mercator projection (meters). Dealing with this kind of conversion is exactly what Mercantile and its like were made to handle.

Tiletanic's use cases are a bit different:

- What do you do if your data is already in some projection? For instance, you might want to know what tile covers a point specified in the Web Mercator projection without first converting back to WGS84. Tiletanic can help.
- At DigitalGlobe, our imagery is often projected into many other projections (UTM, Geographic, etc) and it is often times extremely convienent to organize raster data into a tiled format before proceeding with processing (a single "strip" from one of our satellite collects can easily be 100km long!). When dealing with different projections, the user typically needs to impose their own tiling scheme. Tiletanic provides an easy way for you to define you're own scheme and get a lot of tiling functionality right out of the gate.

CHAPTER 2

User Guide

2.1 Building Blocks

Tiletanic has extremely simple structures (named tuples) for representing the building blocks that compose tiles. You don't have to use these, they just make your life a little easier. Some of the prebaked tiling schemes return these named tuples, but they can be treated as tuples just the same if that's the way you prefer to role.

`Tile`: Just a named tuple for storing (x, y, z) tile coordinates, which is of course the column, row, and zoom level of the tile. Note that the origin of the row and column coordinates is sometimes defined in the bottom left or top left of the grid.

```
>>> from tiletanic import base
>>> tile = base.Tile(1, 2, 3)
>>> tile
Tile(x=1, y=2, z=3)
>>> tile.x
1
>>> tile.y
2
>>> tile.z
3
>>> tile[0]
1
>>> tile[1]
2
>>> tile[2]
3
```

`Coords`: Geospatial coordinate pairs (x, y).

```
>>> base.Coords(0., 1.)
Coords(x=0.0, y=1.0)
```

`CoordsBbox`: Bounding box coordiantes (xmin, ymin, xmax, ymax)

```
>>> base.CoordsBbox(0., 0., 1., 1.)
CoordsBbox(xmin=0.0, ymin=0.0, xmax=1.0, ymax=1.0)
```

2.2 Tiling Schemes

Tile schemes are how you convert back and forth from tile coordinates to geospatial coordinates or quadkeys and the like. They also let you easily traverse the tile structure. You can use one of the schemes that comes with Tiletanic (see `here`) or build your own.

If you build your own, you'll want to implement the public API of a tilescheme (see `here`) so that you can use the tile algorithms defined around this API.

Here's a Web Mercator tile scheme:

```
>>> from tiletanic import tileschemes
>>> tiler = tileschemes.WebMercator()
```

You can check the bounds for which it is defined:

```
>>> tiler.bounds
CoordsBbox(xmin=-20037508.342789244, ymin=-20037508.342789244, xmax=20037508.
↪342789244, ymax=20037508.342789244)
```

Get the XYZ tile coordinates of a geospatial coordinate at a given zoom level:

```
>>> t = tiler.tile(14765187.879790928, -3029352.3049981054, 14)
>>> t
Tile(x=14228, y=9430, z=14)
```

How about that tile's parent and children:

```
>>> tiler.parent(t)
Tile(x=7114, y=4715, z=13)
>>> tiler.children(t)
[Tile(x=28456, y=18860, z=15), Tile(x=28457, y=18860, z=15), Tile(x=28456, y=18861,␣
↪z=15), Tile(x=28457, y=18861, z=15)]
```

What are the upper left, bottom right, and bounding box geospatial coordinates of that tile?

```
>>> tiler.ul(t)
Coords(x=14763964.887338366, y=-3028129.3125455417)
>>> tiler.br(t)
Coords(x=14766410.87224349, y=-3030575.297450669)
>>> tiler.bbox(t)
CoordsBbox(xmin=14763964.887338366, ymin=-3030575.297450669, xmax=14766410.87224349,␣
↪ymax=-3028129.3125455417)
```

Conversion to and from quadkeys is also supported:

```
>>> qk = tiler.quadkey(t)
>>> qk
'31031132030320'
>>> tiler.quadkey_to_tile(qk)
Tile(x=14228, y=9430, z=14)
```

2.3 Tile Covering

Often times, one is given a geometry and would like to know what tiles at a given zoom level cover it. Luckily for you, Tiletanic provides just such functionality! Just define your tile scheme, get a shapely geometry representing the geometry you'd like covered, and call `cover_geometry()`.

Here's an example using the previous output tile Tile(x=14228, y=9430, z=14):

```
>>> from tiletanic import tilecover
>>> from shapely import geometry
>>> [t for t in tilecover.cover_geometry(tiler, geometry.box(*tiler.bbox(t)), 14)]
[Tile(x=14228, y=9430, z=14), Tile(x=14229, y=9430, z=14), Tile(x=14228, y=9431,␣
↪z=14), Tile(x=14229, y=9431, z=14), Tile(x=14230, y=9430, z=14), Tile(x=14230,␣
↪y=9431, z=14), Tile(x=14228, y=9432, z=14), Tile(x=14229, y=9432, z=14),␣
↪Tile(x=14230, y=9432, z=14)]
```

Note that 9 tiles are returned; this is expected as a tile has 8 neighbor tiles that touch it at a given level. If we try a corner tile at that same level, we get back four tiles as expected:

```
>>> [t for t in tilecover.cover_geometry(tiler, geometry.box(*tiler.bbox(0,0,14)),␣
↪14)]
[Tile(x=0, y=0, z=14), Tile(x=1, y=0, z=14), Tile(x=0, y=1, z=14), Tile(x=1, y=1,␣
↪z=14)]
```

`cover_geometry()` works with all the shapely geometry types (Points, Polygons, and LineStrings as well as their Multi versions).

CHAPTER 3

API Documentation

3.1 tiletanic.base module

3.2 tiletanic.tilecover module

3.3 tiletanic.tileschemes module

CHAPTER 4

Command Line Interface

Tiletanic's command line interface is a program named *tiletanic*. Execute *tiletanic –help* for more information.

4.1 cover_geometry

Added in 0.0.5

Given an area of interest, calculate the tile covering at a particular zoom level. Execute *tiletanic cover_geometry –help* for details.