In this lecture, we will discuss…

- Geospatial Introduction
- 2dsphere
- 2dsphere Index and Queries
- `$near`
- `$minDistance`
- `$maxDistance`

Geolocation - Introduction

- **Geo** – refers to Earth
- **Location** – refers to the position on the Earth
- **Geolocation Services** – multi-billion dollar industry
  - Smart Phone, GPS Devices
  - Social Media Apps – local deals, local news, local specials

Geolocation - Introduction

- MongoDB allows you to
  - create, index and query geospatial data
  - geospatial data = location data
- Data stored in spherical surface (`2dsphere` index)

2dsphere Index

- A `2dsphere` index supports queries that calculate geometries on an earth-like sphere
- Default datum for an earth-like sphere is WGS84. Coordinate-axis order is longitude, latitude.
- `2dsphere` index supports all MongoDB geospatial queries:
  - queries for inclusion, intersection, and proximity

Geolocation - GeoJSON

- The `2dsphere` index supports data stored as GeoJSON object – always list coordinates as [longitude, latitude]
  - `{ type: "Point", coordinates: [ 40, 5 ] }`
- GeoJSON Objects
  - Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon, Geometry Collection

2dsphere Index Creation

- `db[:zips].indexes.create_one ({:loc=>Mongo::Index::GEO2DSPHERE})`
- A 2dsphere index was added to the MongoDB collection index.
- `:loc` contains [latitude, longitude]

- *Note: `loc` uses latitude, longitude format. Data format and query syntax must be consistent.*

2dsphere Queries

- `db[:zips].find(:city => 'BALTIMORE').first`
- **Output:** `{"_id"=>"21201", "city"=>"BALTIMORE", "loc"=>[-76.625203, 39.29463], "pop"=>16256, "state"=>"MD"}`
- Let's find some cities close to Baltimore using `$near` using `$minDistance` and `$maxDistance`

2dsphere Queries

```
db[:zips].find(:loc=>
        {:$near=>{
            :$geometry=>{:type=>"Point",:coordinates=>[-76.625203, 39.29463]},
            :$minDistance=>10000,
            :$maxDistance=>50000
          }
        }).limit(5).each { |r| pp r }
```

Summary

- Geospatial indexes and query help in fetch data based on proximity
- Useful with building custom pages that are “geo specific”

What’s Next?

- Demo