Commands

MongoDB provides statistics about your collections but you need to call the actual commands. For database wide statistics call the `dbstats` command:

In [1]:
```
import pymongo
conn = pymongo.Connection()
conn.drop_database('example')

db = conn.example
db.test.insert({'hello': 'world'})

db.command({'dbstats': 1})
```

Out[1]:
```
{u'avgObjSize': 42.399999999999999,
 u'collections': 3,
 u'dataSize': 212,
 u'db': u'example',
 u'fileSize': 201326592,
 u'indexSize': 8176,
 u'indexes': 1,
 u'nsSizeMB': 16,
 u'numExtents': 3,
 u'objects': 5,
 u'ok': 1.0,
 u'storageSize': 12288}
```

To get collection statistics use the `collstats` command and pass in the collection name that you want stats for.

In [2]:
```
db.command({'collstats': 'test'})
```

Out[2]:
```
{u'avgObjSize': 40.0,
 u'count': 1,
 u'indexSizes': {u'_id_': 8176},
 u'lastExtentSize': 4096,
 u'nindexes': 1,
 u'ns': u'example.test',
 u'numExtents': 1,
 u'ok': 1.0,
 u'paddingFactor': 1.0,
 u'size': 40,
 u'storageSize': 4096,
 u'systemFlags': 1,
 u'totalIndexSize': 8176,
 u'userFlags': 0}
```

Capped collections

Capped collections are fixed sized collections that have a very high performance auto-FIFO age-out feature (age out is based on insertion order).

In [3]:
```
db.create_collection('capped', capped=True, size=1000)

for i in xrange(1, 1000):
```

```
    db.capped.insert({'name': "%s" % i, "value": i})

db.capped.count()
```

Out[3]: 59

Tailable cursors

A tailable cursor "tails" the end of a *capped collection*, much like the Unix "tail -f" command. They key idea is that if we "catch up" and have reached the end of the collection, our position is remembered rather than the cursor being closed. Thus, after new objects are inserted, we can resume retrieving from where we left off – which is then very inexpensive. If you reach the end of the collection and request more data, the cursor will block waiting for new documents to be inserted.

In [4]:
```
run = False #  Done so I dont break Ipython notebook
import time

cursor = db.capped.find(tailable=True)
while cursor.alive and run:
    try:
        doc = cursor.next()
        print doc
    except StopIteration:
        time.sleep(1)
```

Exercises

I. List all the collections in the `example` database
II. After loading data into the `capped` collection what is its size?
III. Why are there only 59 documents in the `capped` collection?
IV. List the indexes for the `capped` collection
V. Test the tailable cursor by creating a tailable cursor in one shell and in another add some data
VI. What happens if you try to create a tailable cursor on the `test` collection?