Couchbase

Couchbase Server 1.8.1 Java Client Library

Cheat Sheet

Connection Management

Connecting to Couchbase Server

private static **CouchbaseClient client**;
client.new CouchbaseClient([**url**] [, **urls**] [, **username**] [, **password**])

Parameters:

String url	URL for Couchbase Server Instance, or node
String urls	Linked list containing one or more URLs as strings
String username	Username for Couchbase bucket
String Password	Password for Couchbase bucket

Disconnecting from Couchbase Server

client.shutdown();
client.shutdown(**timevalue**, **timeunit**);

Parameters:

long timevalue	Wait value until any outstanding queued work is completed
timeunit	TimeUnit.SECONDS, TImeUnit.MINUTES

Store Operations

Add Operations

The add method adds a value to the database with the specified key, but will fail if the key already exists in the database.

client.add(**key**, **expiry**, **value**)
client.add(**key**, **expiry**, **value**, **transcoder**)

Set Operations

The set method stores a value to the database with a specified key, even if the key already exists and has data. This operation overwrites the existing value with new data.

client.set(**key**, **expiry**, **value**)
client.set(**key**, **expiry**, **value**, **transcoder**)

Parameters:

String key	Key used to reference the value. The key cannot contain control characters or whitespace
int expiry	Expiry time for the key in seconds. Values larger than 30*24*60*60 seconds (30 days) are interpreted as absolute times (from the epoch)
Object value	Value to be stored
Transcoder<T> transcoder	Transcoder class to be used to serialize values

Retrieve Operations

There are several different flavors of retrieve operations in the Couchbase Server 1.8.1 SDK.

client.asyncGetAndTouch(**key**, **expiry**)
client.asyncGetAndTouch(**key**, **expiry**, **transcoder**)
client.getAndTouch(**key**, **expiry**)
client.getAndTouch(**key**, **expiry**, **transcoder**)
client.asyncGet(**key**)
client.asyncGetBulk(**keycollection**)
client.asyncGetBulk(**keyn**)
client.asyncGetBulk(**transcoder**, **keyn**)
client.asyncGetBulk(**keycollection**, **transcoder**)
client.asyncGet(**key**, **transcoder**)
client.get(**key**)
client.getBulk(**keycollection**)
client.getBulk(**keyn**)
client.getBulk(**transcoder**, **keyn**)
client.getBulk(**keycollection**, **transcoder**)
client.get(**key**, **transcoder**)
client.getAndLock(**key**, **expiry**, **transcoder**)
client.getAndLock(**key**, **expiry**, **transcoder**)
client.asyncGets(**key**)
client.asyncGets(**key**, **transcoder**)
client.gets(**key**)
client.gets(**key**, **transcoder**)
client.unlock(**key**, **casunique**)

Parameters:

String key	Key used to reference the value. The key cannot contain control characters or whitespace
int expiry	Expiry time for the key in seconds. Values larger than 30*24*60*60 seconds (30 days) are interpreted as absolute times (from the epoch)
Collection<String> keycollection	One or more keys used to reference a value
String… keyn	One or more keys used to reference a value
Long casunique	Unique value used to identify a key/value combination
Transcoder<T> transcoder	Transcoder class to be used to serialize values

Statistics Operations

Obtain stats from all servers defined in a CouchbaseClient object

client.getStats()
client.getStats(**statname**)

Parameters:
String statname : Group name of the statistic for selecting individual statistic value

Couchbase

Couchbase Server 1.8.1 Java Client Library

Cheat Sheet

Update Operations

The update methods support different methods of updating and changing existing information within Couchbase Server.

client.**append**(**casunique**, **key**, **value**)
client.**append**(**casunique**, **key**, **value**, **transcoder**)
client.**asyncCAS**(**key**, **casunique**, **value**)
client.**asyncCAS**(**key**, **casunique**, **expiry**, **value**, **transcoder**)
client.**asyncCAS**(**key**, **casunique**, **value**, **transcoder**)
client.**cas**(**key**, **casunique**, **value**)
client.**cas**(**key**, **casunique**, **expiry**, **value**, **transcoder**)
client.**cas**(**key**, **casunique**, **value**, **transcoder**)
client.**asyncDecr**(**key**, **offset**)
client.**decr**(**key**, **offset**, default)
client.**decr**(**key**, **offset**, default, **expiry**)
client.**delete**(**key**)
client.**asyncIncr**(**key**, **offset**)
client.**incr**(**key**, **offset**)
client.**incr**(**key**, **offset**, default)
client.**incr**(**key**, **offset**, default, **expiry**)
client.**prepend**(**casunique**, **key**, **value**)
client.**prepend**(**casunique**, **key**, **value**, **transcoder**)
client.**touch**(**key**, **expiry**)

Parameters:

long casunique	Unique value used to identify a key/value combination
String key	Key used to reference the value. The key cannot contain control characters or whitespace
int offset	Integer offset value to increment / decrement (default is 1)
Int default	Default value to increment/decrement if key does not exist
int expiry	Expiry time for the key in seconds. Values larger than 30*24*60*60 seconds (30 days) are interpreted as absolute times (from the epoch)
Object value	Value to be stored
Transcoder<T> transcoder	Transcoder class to be used to serialize values

Useful Links:

Couchbase Website : http://www.couchbase.com

Couchbase Blog : http://blog.couchbase.com

Github Source : http://github.com/couchbaselabs

Twitter : #couchbase

Quick Troubleshooting Tips

1. Configuring Logging

You can configure logging using the following options –

(a) Use spymemcached to log from the Java SDK and set your JDK properties to log Couchbase Java SDK information

(b) Setup logging from within your application

2. Handling Timeouts

The Java library has a set of synchronous and asynchronous methods. While it does not happen in most situations, occasionally network I/O can become congested, nodes can fail, or memory pressure can lead to situations where an operation can timeout.

(a) Synchronous methods on the clients will return a RuntimeException showing timeout as the root cause.

(b) Asynchronous methods will throw a TimeoutException

Retry the operation in the form of a backoff or exponential backoff

3. Timing-out and Blocking

You can choose to control the volume of asynchronous requests that are issued by your application by setting a timeout for blocking.

```
<URI> baselist = new ArrayList<URI>();
    baselist.add(new URI("http://localhost:8091/pools"));

CouchbaseConnectionFactoryBuilder cfb = new
CouchbaseConnectionFactoryBuilder();

// wait up to 5 seconds when trying to enqueue an operation
cfb.setOpQueueMaxBlockTime(5000);

 CouchbaseClient myclient = new
CouchbaseClient(cfb.buildCouchbaseConnection(baselist, "default",
"default", ""));
```

4. Bulk Load and Exponential Backoff

When you bulk load data to Couchbase Server, you can accidentally overwhelm available memory in the Couchbase cluster before it can store data on disk. If this happens, Couchbase Server will immediately send a response indicating the operation cannot be handled at the moment but can be handled later. This is sometimes referred to as "handling Temp OOM", where OOM means out of memory. As shown below, exponential backoff can be used in such cases to retry

```
while(backoffexp < MAX_RETRIES) {
      double backoffMillis = Math.pow(2, backoffexp);
      backoffMillis = Math.min(1000, backoffMillis); // 1 sec max
      Thread.sleep((int) backoffMillis);
      System.err.println("Backing off, tries so far: " + backoffexp);
     // Retry Action (break out of loop after success)
     backoffexp++;
}
```