http://is.gd/jaxwkshp

#JAXLondon

A Fluent API for MongoDB

Trisha Gee, Java Driver Developer

@trisha_gee

Enough about me, what about you?

mongoDB

Our Domain

mongoDB

MongoDB

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Installation

Getting Started

http://is.gd/HkQqOF

Copy the contents of the **workshop** directory on the USB stick into a new work area. I've put mine in ~/Documents/workshops/jax. For the rest of these instructions I'll refer to it as **<location>**.

Download MongoDB or get it off the USB stick (install/mongodb)

Unzip into **<location>/mongodb**

To start MongoDB:

cd <location>

./mongodb/bin/mongod --dbpath data

Now MongoDB should be running on localhost and port 27017

(optional) ./mongodb/bin/mongo --port 27017

What's the problem?

mongoDB

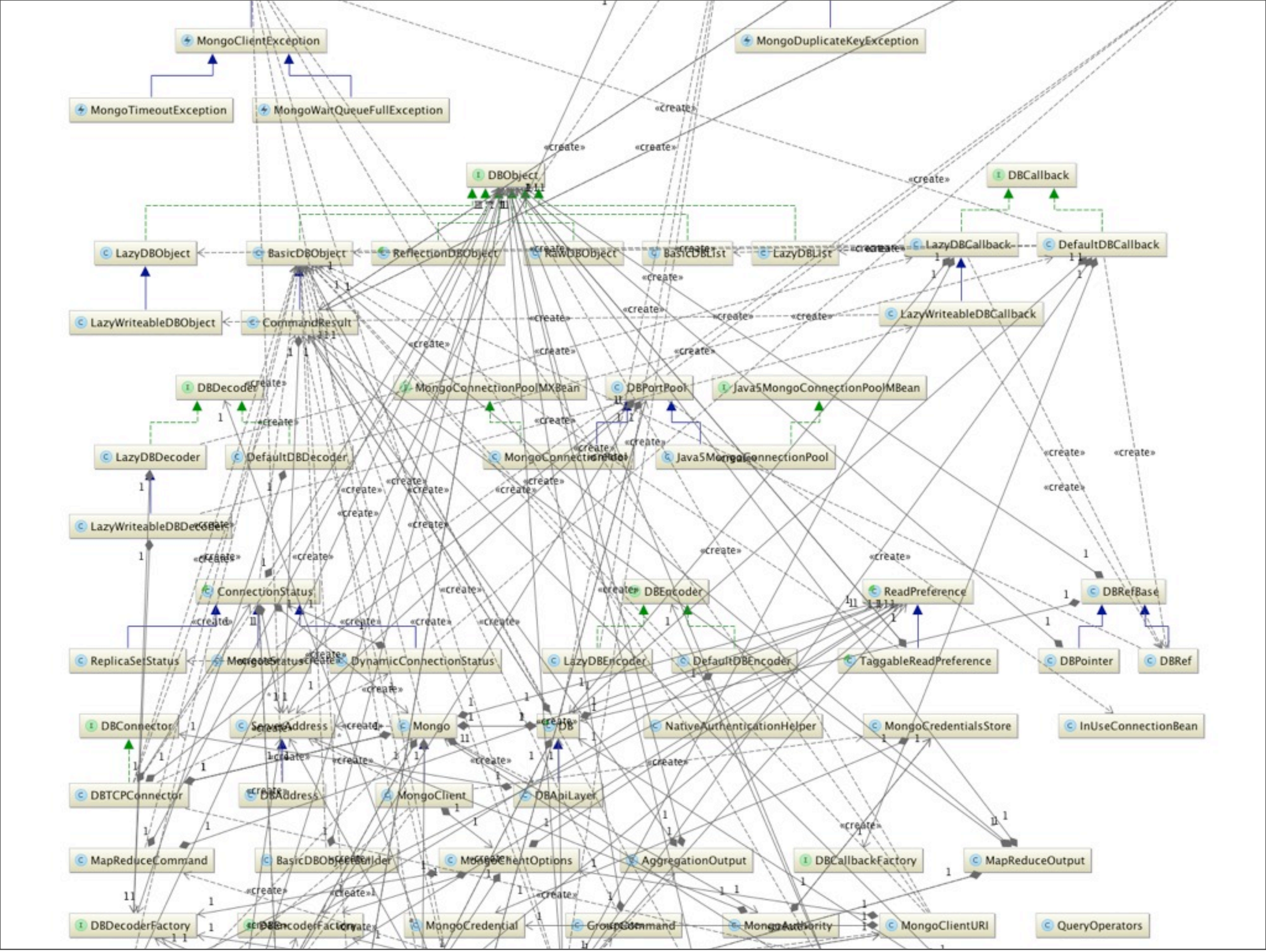
The Driver

```
collection.find
```

find(DBObject ref)	DBCursor
find()	DBCursor
find(DBObject ref, DBObject ke…	DBCursor
~~find(DBObject query, DBObject …~~	DBCursor
~~find(DBObject query, DBObject …~~	DBCursor
findAndModify(DBObject query, …	DBObject
findOne()	DBObject
findAndModify(DBObject query, …	DBObject
findAndModify(DBObject query, …	DBObject
findAndRemove(DBObject query)	DBObject
findOne(DBObject o)	DBObject

Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)

```
208     * @param coll
209     * @param m
210     * @param hostNeeded
211     * @param readPref
212     * @param decoder
213     * @return
214     * @throws MongoException
215     */
216    @Override
217    public Response call( DB db, DBCollection coll, OutMessage m, ServerAddress hostNeeded, int retries,
218                          ReadPreference readPref, DBDecoder decoder ){
219        try {
220            return innerCall(db, coll, m, hostNeeded, retries, readPref, decoder);
221        } finally {
222            m.doneWithMessage();
223        }
224    }
225
226    // This method is recursive.  It calls itself to implement query retry logic.
227    private Response innerCall(final DB db, final DBCollection coll, final OutMessage m, final ServerAddress hostNeeded,
228                               final int retries, ReadPreference readPref, final DBDecoder decoder) {
229        if (readPref == null)
230            readPref = ReadPreference.primary();
231
232        if (readPref == ReadPreference.primary() && m.hasOption( Bytes.QUERYOPTION_SLAVEOK ))
233            readPref = ReadPreference.secondaryPreferred();
234
235        boolean secondaryOk = !(readPref == ReadPreference.primary());
236
237        _checkClosed();
238        // Don't check master on secondary reads unless connected to a replica set
239        if (!secondaryOk || getReplicaSetStatus() == null)
240            checkMaster( false, !secondaryOk );
241
242        final DBPort port = _myPort.get(false, readPref, hostNeeded);
243
244        Response res = null;
245        boolean retry = false;
246        try {
247            port.checkAuth( db.getMongo() );
248            res = port.call( m , coll, decoder );
249            if ( res._responseTo != m.getId() )
250                throw new MongoException( "ids don't match" );
251        }
252        catch ( IOException ioe ){
253            _myPort.error(port, ioe);
254            retry = retries > 0 && !coll._name.equals( "$cmd" )
255                    && !(ioe instanceof SocketTimeoutException) && _error( ioe, secondaryOk );
256            if ( !retry ){
257                throw  new MongoException.Network("Read operation to server " + port.host() + " failed on database " + db , ioe );
258            }
259        }
260        catch ( RuntimeException re ){
261            _myPort.error(port, re);
262            throw re;
263        } finally {
264            _myPort.done(port);
265        }
266
267        if (retry)
268            return innerCall( db , coll , m , hostNeeded , retries - 1 , readPref, decoder );
269
270        ServerError err = res.getError();
271
272        if ( err != null && err.isNotMasterError() ){
273            checkMaster( true , true );
274            if ( retries <= 0 ){
275                throw new MongoException( "not talking to master and retries used up" );
276            }
277            return innerCall( db , coll , m , hostNeeded , retries -1, readPref, decoder );
278        }
279
280        return res;
281    }
282
```


MongoClientException
MongoDuplicateKeyException
MongoTimeoutException
MongoWaitQueueFullException
DBObject
DBCallback
LazyDBObject
BasicDBObject
ReflectionDBObject
RawDBObject
BasicDBList
LazyDBList
LazyDBCallback
DefaultDBCallback
LazyWriteableDBObject
CommandResult
LazyWriteableDBCallback
DBDecoder
MongoConnectionPoolMXBean
DBPortPool
Java5MongoConnectionPoolMBean
LazyDBDecoder
DefaultDBDecoder
MongoConnectionPool
Java5MongoConnectionPool
LazyWriteableDBDecoder
ConnectionStatus
DBEncoder
ReadPreference
DBRefBase
ReplicaSetStatus
MongoStatus
DynamicConnectionStatus
LazyDBEncoder
DefaultDBEncoder
TaggableReadPreference
DBPointer
DBRef
DBConnector
ServerAddress
Mongo
DB
NativeAuthenticationHelper
MongoCredentialsStore
InUseConnectionBean
DBTCPConnector
DBAddress
MongoClient
DBApiLayer
MapReduceCommand
BasicDBObjectBuilder
MongoClientOptions
AggregationOutput
DBCallbackFactory
MapReduceOutput
DBDecoderFactory
DBEncoderFactory
MongoCredential
GroupCommand
MongoAuthority
MongoClientURI
QueryOperators
«create»

Why is this my problem?

Retrospective

mongoDB

What do we want to do?

mongoDB

Design Goals

- Intuitive API
- Consistency
- Understandable exceptions
- Cleaner design
- Test friendly
- Backwards compatible
- http://is.gd/java3mongodb

Happy Users

Three Types Of Users

1. Java Developers
2. ODMs / other drivers / third parties
3. Contributors

Setting up the IDE

Extract the Java project into **<location>/java-project/java3.0**

```
cd <location>/java-project/java3.0
```

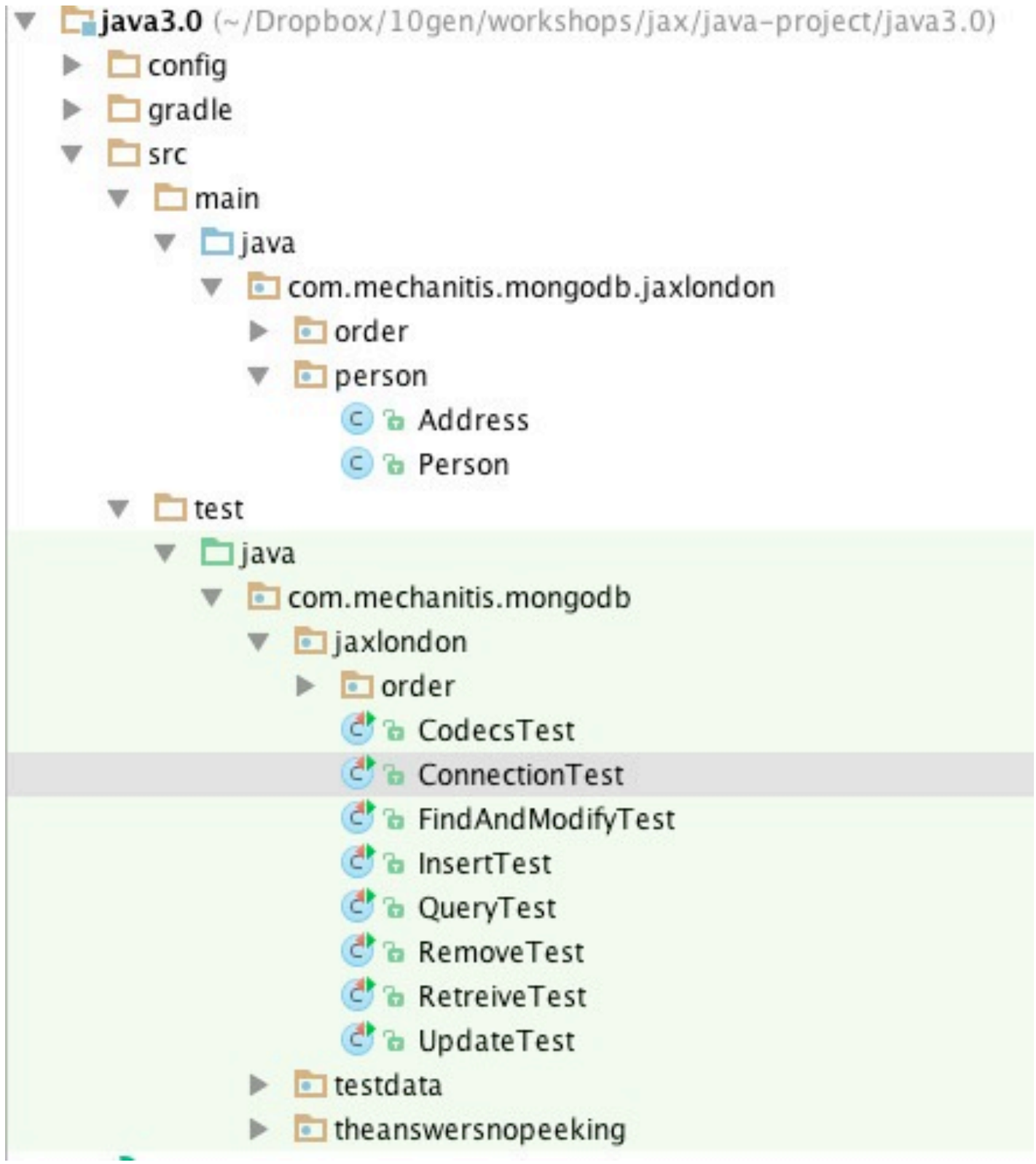
Run:

```
gradle idea
```

or

```
gradle eclipse
```

Open in your favourite IDE and you should be ready to start playing along


java3.0 (~/Dropbox/10gen/workshops/jax/java-project/java3.0)
config
gradle
src
main
java
com.mechanitis.mongodb.jaxlondon
order
person
Address
Person
test
java
com.mechanitis.mongodb
jaxlondon
order
CodecsTest
ConnectionTest
FindAndModifyTest
InsertTest
QueryTest
RemoveTest
RetreiveTest
UpdateTest
testdata
theanswersnopeeking

```
MongoClient mongoClient =
    MongoClients.create(new MongoClientURI("mongodb://localhost:27017"));

MongoDatabase myDatabase = mongoClient.getDatabase("myDatabase");

MongoCollection<Document> myCollection =
    myDatabase.getCollection("myCollection");
```

Connecting

ConnectionTest

The New API

mongoDB

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Caveats

- It won't look like this
- Haven't decided consistent names yet
- Need something that suits all drivers
- Do Not Use In Production Yet!

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Documents

- What does a document look like?
- How do I create one in Java?
- How do I get them back out again?

```
patron = {
  _id: "joe",
  name: "Joe Bookreader",
  address: {
    street: "123 Fake St",
    city: "Faketon",
    state: "MA",
    zip: 12345
  }
  books: [ 27464, 747854, ...]
}
```

Creating a Document

```
List books = new BasicDBList();
books.add(27464, 747854);
new BasicDBObject("_id", "joe")
          .append("name", "Joe Bookreader")
          .append("address", new BasicDBObject("street", "123 Fake St")
                                      .append("city", "Faketon")
                                      .append("state", "MA")
                                      .append("zip", 12345))
          .append("books", books);

collection.insert(books);
```

Building a Document

```
List books = new ArrayList();
books.add(27464, 747854);
new Document("_id", "joe")
    .append("name", "Joe Bookreader")
    .append("address", new Document("street", "123 Fake St")
                            .append("city", "Faketon")
                            .append("state", "MA")
                            .append("zip", 12345))
    .append("books", books);

collection.insert(books);
```

Building a Document

Create a Document

InsertTest.shouldTurnAPersonIntoADocument

Saving

```
InsertTest.shouldBeAbleToSaveAPerson
```

Retrieving a Document

```java
DBCollection collection =
    database.getCollection("coll");

final ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

DBObject query = new BasicDBObject("name", theNameToFind);
DBCursor results = collection.find(query);
for (DBObject dbObject : results) {
    DBObject addressDocument = (DBObject) dbObject.get("address");
    Patron patron = new Patron((String) dbObject.get("name"),
                        new Address((String)dbObject.get("street"),
                                    (String)addressDocument.get("city"),
                                    (String)addressDocument.get("state"),
                                    (Integer)addressDocument.get("zip")),
                        (BasicDBList)dbObject.get("books"));
    resultsToReturn.add(patron);
}
return resultsToReturn;
```

Getting it back

```java
MongoCollection<Document> collection =
    database.getCollection("coll");

final ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

Document query = new Document("name", theNameToFind);
collection.find(query).forEach(new Block<Document>() {
    public boolean run(Document document) {
        Document addressDocument = (Document)document.get("address");
        Patron patron = new Patron((String)document.get("name"),
                                   new Address((String)addressDocument.get("street"),
                                               (String)addressDocument.get("city"),
                                               (String)addressDocument.get("state"),
                                               (int)addressDocument.get("zip")),
                                   ((List<Integer>)document.get("books")));
        return resultsToReturn.add(patron);
    }
});
return resultsToReturn;
```

New Document Type

```
MongoCollection<Document> collection =
    database.getCollection("coll");

final ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

Document query = new Document("name", theNameToFind);
collection.find(query).forEach(new Block<Document>() {
    public boolean run(Document document) {
        Document addressDocument = (Document)document.get("address");
        Patron patron = new Patron((String)document.get("name"),
                            new Address((String)addressDocument.get("street"),
                                        (String)addressDocument.get("city"),
                                        (String)addressDocument.get("state"),
                                        (int)addressDocument.get("zip")),
                            ((List<Integer>)document.get("books")));
        return resultsToReturn.add(patron);
    }
});
return resultsToReturn;
```

```
MongoCollection<Document> collection =
    database.getCollection("coll");

ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

Document query = new Document("name", theNameToFind);
collection.find(query).forEach(document -> {

        Document addressDocument = (Document)document.get("address");
        Patron patron = new Patron((String)document.get("name"),
                            new Address((String)addressDocument.get("street"),
                                        (String)addressDocument.get("city"),
                                        (String)addressDocument.get("state"),
                                        (int)addressDocument.get("zip")),
                            ((List<Integer>)document.get("books")));
        return resultsToReturn.add(patron);

});
return resultsToReturn;
```

Retrieving

RetrieveTest

Options

- Transformation handled in correct layer
- Use an ODM to do it for you:
  - Morphia
  - Spring Data
  - MongoJack
  - etc
- Provide your own Codec

```
MongoCollection<Patron> collection =
    database.getCollection("coll", new PatronCodec());



Document query = new Document("name", theNameToFind);



return collection.find(query).into(new ArrayList<Patron>());
```

Better still...

```
MongoCollection<Patron> collection =
    database.getCollection("coll", new PatronCodec());



Document query = new Document("name", theNameToFind);



return collection.find(query).into(new ArrayList<Patron>());
```

Separation of concerns

Codecs

CodecsTest

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Read

```
collection.find(query).skip(1000).limit(100);
collection.find(query).skip(1000).limit(100);
```

Find

```
collection.find(query).skip(1000).limit(100);
collection.find(query).skip(1000).limit(100);

collection.find(query, fields);
```

Find

```
collection.find
```

find(DBObject ref)	DBCursor
find()	DBCursor
find(DBObject ref, DBObject ke…	DBCursor
~~find(DBObject query, DBObject …~~	DBCursor
~~find(DBObject query, DBObject …~~	DBCursor
findAndModify(DBObject query, …	DBObject
findOne()	DBObject
findAndModify(DBObject query, …	DBObject
findAndModify(DBObject query, …	DBObject
findAndRemove(DBObject query)	DBObject
findOne(DBObject o)	DBObject

Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)

Which One?

```
collection.find(query).skip(1000).limit(100);
collection.find(query).skip(1000).limit(100);

collection.find(query, fields);
collection.find(query).fields(fields);
```

Find

```
collection.find
    find(ConvertibleToDocument filter)   MongoView<Document>
    find()                               MongoView<Document>
    find(Document filt…                  MongoView<Document>
```

Fewer Decisions

```
collection.find(query).
```

```
project(Document selector)   MongoView<Document>
into(A target)                                   A
find(ConvertibleToD…   MongoView<Document>
find(Document filte…   MongoView<Document>
limit(int limit)       MongoView<Document>
project(Convertible…   MongoView<Document>
skip(int skip)         MongoView<Document>
sort(ConvertibleToD…   MongoView<Document>
sort(Document sortC…   MongoView<Document>
upsert()               MongoView<Document>
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)
```

“Ctrl + space” friendly

Querying

QueryTest

Delete

```
collection.remove(query);

collection.find(query).remove();
```

Remove

Removing

RemoveTest

Update

```
collection.update(query, newValues)

collection.find(query).updateOne(newValues);
```

Update

```
collection.upd
    update(DBObject q, DBObjec…   WriteResult
    update(DBObject q, DBObjec…   WriteResult
    update(DBObject q, DBObjec…   WriteResult
    update(DBObject q, DBObject o, boolean upsert, boolean multi,
    updateMulti(DBObject q, DB…   WriteResult
```

Overloaded Methods

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);
```

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);


collection.update(query, newValues, true, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .upsert()
          .updateOne(newValues);
```

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);


collection.update(query, newValues, true, true, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .upsert()
          .update(newValues);
```

Update

Update

UpdateTest

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Atomic Operations

```
collection.findAndModify(query, newValues);

collection.find(query)
          .getOneAndUpdate(newValues);
```

Find and Modify

```
collection.findAnd
    findAndModify(DBObject query, DBObject fields, DBObject sort, boolean remove, DBObject update, boolean returnNew, boolean upsert)  DBObject
    findAndModify(DBObject query,…  DBObject
    findAndModify(DBObject query,…  DBObject
    findAndRemove(DBObject query)   DBObject
    Dot, semicolon and some other keys will also close this lookup and be inserted into editor
```

They hate me!

```
collection.findAndModify(query, update);

collection.find(query)
          .getOneAndUpdate(update);


collection.findAndModify(query,
                         fields,
                         criteria,
                         false,
                         newValues,
                         false,
                         false);

collection.find(query)
          .project(fields)
          .sort(criteria)
          .getOneAndUpdate(newValues);
```

Find and Modify

```
collection.findAndModify(query, update);

collection.find(query)
          .getOneAndUpdate(update);


collection.findAndModify(query,
                         fields,
                         criteria,
                         false,
                         newValues,
                         true,
                         false);

collection.find(query)
          .project(fields)
          .sort(criteria)
          .updateOneAndGet(newValues);
```

Find and Modify

Find and Modify

FindAndModifyTest

Consistency

```
collection.find(query).count();
collection.find(query).remove();
collection.find(query).update(newValues);
collection.find(query).updateOneAndGet(newValues);
collection.find(query).getOneAndUpdate(newValues);

collection.find(query).sort(sortCriteria).skip(9).limit(10).get();
collection.find(query).sort(sortCriteria).skip(9).limit(10).getOne();

collection.find(query).sort(ascending("name")).getOne();
```

Consistency at last

Playtime

In Summary

mongoDB

- New API has similarities to the old one
- ...but is designed with consistency in mind
- DSL-style chaining of operations
- ...with terminators to instigate the command
- Codecs & generics to work with Real Objects
- Still plenty of work to do

Also...

mongoDB

Logical Exception Handling

mongoDB

- Client Exceptions
- Server Exceptions
- No more parsing error Strings

Test Friendly

mongoDB

- Interfaces!
- Acceptance, functional and unit tests

And Documentation...

mongoDB

- Self documenting code
- JavaDoc
- Unit, Functional and Acceptance Tests
- Blogs

Backwards compatible

trisha.gee@mongodb.com
@trisha_gee

http://is.gd/java3mongodb

mongoDB

Atomic Operators

mongoDB

- $inc - increment a particular value by a certain amount
- $set - set a particular value
- $unset - delete a particular field (v1.3+)
- $push - append a value to an array
- $pushAll - append several values to an array
- $addToSet - adds value to the array only if its not in the array already
- $pop - removes the last element in an array
- $pull - remove a value(s) from an existing array
- $pullAll - remove several value(s) from an existing array
- $rename - renames the field
- $bit - bitwise operations

Tools

- Create & drop indexes
- Create & rename collections
- Ping & other commands

Other Commands

http://is.gd/mongocmds

Not covered

- Authentication
- Aggregation