**MongoDB3.0的Java Driver介绍**

目录

[1. 驱动下载 2](#_Toc420997865)

[2. 使用介绍 2](#_Toc420997866)

[2.1. 获取连接 2](#_Toc420997867)

[2.2. 获取一个集合 3](#_Toc420997868)

[2.3. 插入一条文档 3](#_Toc420997869)

[2.4. 查询所有文档 3](#_Toc420997870)

[2.5. 查询单一文档 3](#_Toc420997871)

[2.6. 更新文档 4](#_Toc420997872)

[2.7. 删除 4](#_Toc420997873)

[3. API文档 5](#_Toc420997874)

# 驱动下载

从 http://central.maven.org/maven2/org/mongodb/mongo-java-driver/选择一个版本进行下载，目前最新的驱动版本是3.0.2，具体下载以下jar包：

[mongo-java-driver-3.0.2.jar](http://central.maven.org/maven2/org/mongodb/mongo-java-driver/3.0.2/mongo-java-driver-3.0.2.jar)

[mongo-java-driver-3.0.2-sources.jar](http://central.maven.org/maven2/org/mongodb/mongo-java-driver/3.0.2/mongo-java-driver-3.0.2-sources.jar)

# 使用介绍

使用MongoDB的Java驱动非常简单，只要确保将mongo-java-driver-3.0.2.jar文件添加到classpath中即可。

## 获取连接

使用类com.mongodb.MongoClient来链接运行有mongod进程的MongoDB实例，并通过 com.mongodb.client.MongoDatabase 接口来访问MongoDB数据库。访问的过程中需要指定要连接的数据库如果库不存在，MongoDB将创建一个新的库。另外，在连接时需要指定服务器地址和端口。下面是使用不同方式连接本地机器的UFS数据库的方式：

MongoClient mongoClient = **new** MongoClient();

MongoClient mongoClient = **new** MongoClient("localhost");

MongoClient mongoClient = **new** MongoClient("localhost", 27017);

MongoClient mongoClient = **new** MongoClient(Arrays.*asList*(**new** ServerAddress("localhost", 27017)));

MongoDatabase db = mongoClient.getDatabase("UFS");

## 获取一个集合

获取一个集合，只需要指定集合名称调用getCollection(Sring collectionName)方法：

MongoCollection<Document> collection = db.getCollection("default");

一旦取得集合对象，就可以对其进行查询，插入数据等操作。

## 插入一条文档

一旦取得集合对象，就可以在集合中插入文档。例如，一个json格式表示的文档(下面文档中有一个内嵌文档)：

{file" : "FileInputStream", "type" : "FileType", "version" : 1, "extInfo" : {"size": 122, "module": "default"}}

使用Document类来创建文档(包括内部文档)，然后调用集合的insertOne()方法插入文档：

db.getCollection("default").insertOne(

**new** Document("file", "FileInputStream").append("type", "FileType").append("version", 1)

.append("extInfo", **new** Document("size", 122).append("module", "default")));

## 查询所有文档

调用不带有参数的find()方法可以返回一个集合中的所有文档，例如以下的查询返回的是集合default中的所有文档：

FindIterable<Document> iterable = db.getCollection("default").find();

find()返回的是FindIterable对象，对该对象遍历可输出所有的文档信息：

iterable.forEach(**new** Block<Document>() {

@Override

**public** **void** apply(**final** Document document) {

System.***out***.println(document);

}

});

## 查询单一文档

可以通过给find()方法传递一个参数来查询集合的部分文档。例如，查出文档的"\_id"字段的值为"556aa95ba990f0060cea28a4"的文档：

FindIterable<Document> iterable =

db.getCollection("default").find(**new** Document("\_id", "556aa95ba990f0060cea28a4"));

iterable.forEach(**new** Block<Document>() {

@Override

**public** **void** apply(**final** Document document) {

System.***out***.println(document);

}

});

以上的查询结果将打印出一条文档：

Document{{\_id=556aa95ba990f0060cea28a4, [content=org.bson.types.Binary@d4c9f081}](mailto:content=org.bson.types.Binary@d4c9f081%7d)}

也可以使用Filters指定查询的条件， 如：

FindIterable<Document> iterable = db.getCollection("default").find(*eq*("\_id", "556aa95ba990f0060cea28a4"));

## 更新文档

为了修改文档中的key对对应的value值，MongoDB提供了update操作， 如使用$set来修改values。使用$set更新操作时，如果key不存在，则会新增key, value值。

以下的更新操作是更新\_id值为” 556aa95ba990f0060cea28a4”所对应的文档中的content值。

db.getCollection("default").updateOne(**new** Document("\_id", "556aa95ba990f0060cea28a4"),

**new** Document("$set", **new** Document("content", "East 31st Street")));

updateOne操作会返回包含操作信息的UpdateResult对象，调用该对象中的getModifiedCpunt()方法可以得到修改的文档数目。

## 删除

删除所有content值为“Manhattan”的文档：

db.getCollection("default").deleteMany(**new** Document("content", "Manhattan"));

deleteMany会返回DeleteResult对象， 使用getDeletedCount()方法可以获得删除文档的数目。

要删除一个集合中的所有文档，可以给deleteMany()传递一个不带有参数的document{}

db.getCollection("default").deleteMany(**new** Document());

删除的操作只会从集合中删除文档， 集合本身包括集合创建的索引还保留。删除整个集合要比删除集合中的所有文档更高效，使用drop()方法可以删除一个集合，包括其中的任何索引。

db.getCollection("default").drop();

# API文档

相关的API文档可以访问：http://api.mongodb.org/java/3.0/