mongodb java api

2010-11-05 09:11:57| 分类： nosql | 标签：dbobject mongodb coll cursor 文档 |字号 订阅

该文档是翻译自文档[mongodb-docs-2010-10-24.pdf]的[Java Language Center]章节，根据自己的理解整理而成。

希望能给像我这样开始接触的朋友一点帮助，同时也做个备忘，因为是刚刚学习，其中的很多功能目前都用不上，以后万一有什么功能不太清楚，也可以直接查阅该文档了。

MongoDB Java Driver 简单操作

一、Java 驱动一致性

MongoDB 的 Java 驱动是线程安全的，对于一般的应用，只要一个 Mongo 实例即可，Mongo 有个内置的连接池（池大小默认为10个）。

对于有大量写和读的环境中，为了确保在一个 Session 中使用同一个 DB 时，我们可以用以下方式保证一致性：

```
DB mdb = mongo.getDB('dbname');

mdb.requestStart();
//
// 业务代码
//
mdb.requestDone();
```

DB 和 DBCollection 是绝对线程安全的，它们被缓存起来了，所以在应用中取到的可能是同一个对象。

二、保存/查找对象(DBObject)

Java 驱动提供了 DBObject 接口，方便我们保存对象到数据库中。

定义需要保存的对象：

```
public class Tweet implements DBObject {
  /** ..... */
}
```

然后我们可以使用该对象：

```
Tweet tweet = new Tweet();
tweet.put("user", userId);
tweet.put("message", message);
tweet.put("date", new Date());

collection.insert(tweet);
```

当从数据库中查询时，结果会自动的转换成 DBObject 对象，我们可以转换成我们自己的类型：

```
collection.setObjectClass(Tweet);

Tweet myTweet = (Tweet)collection.findOne();
```

三、创建连接

```
Mongo m = new Mongo();
Mongo m = new Mongo("localhost");
Mongo m = new Mongo("localhost", 27017);

DB db = m.getDB("mydb);
```

注意：事实上，Mongo 实例代表了一个数据库连接池，即使在多线程的环境中，一个 Mongo 实例对我们来说已经足够了。

四、认证（可选的）

```
boolean auth = db.authenticate("myUserName", "myPasswd");
```

五、取得 Collection 列表

```
Set<String> colls = db.getCollectionNames();

for(String s : colls) {
  System.out.prinln(s);
}
```

六、获取一个 Collection

```
DBCollection coll = db.getCollection("testCollection");
```

使用 DBCollection，我们可以进行插入、查询数据等数据操作。

七、插入文档

假设有个 JSON 文档如下所示：

```
{
  "name": "MongoDB",
  "type": "database",
  "count": 1,
  "info": {
       x: 203,
       y: 102
     }
}
```

注意：上面的 JSON 文档有个内嵌文档"info"。

我们完全可以利用 BasicDBObject 来创建一个和上面的 JSON 一样的文档，并且把它保存在 MongoDB 中。

```
DBObject doc = new BasicDBObject();

doc.put("name", "MongoDB");
doc.put("type", "database");
doc.put("count", 1);

DBObject info = new BasicDBObject();
info.put("x", 203);
info.put("y", 102);

doc.put("info", info);

coll.insert(doc);
```

八、查询第一个文档（**findOne()**）

为了验证在上面我们保存的类似 JSON 的数据，我们可以用 findOne()方法取得数据。

findOne(): 返回一个文档；
find(): 返回一个游标（DBCursor），其中包含一组对象 DBObject；

```
DBObject doc = coll.findOne();
System.out.println(doc);
```

我们将会看到控制台输出：

{ "_id" : "49902cde5162504500b45c2c" , "name" : "MongoDB" , "type" : "database" , "count" : 1 , "info" : { "x" : 203 , "y" : 102} , "_ns" : "testCollection"}

九、插入多个文档

为了在后来展示更多的查询方法，我们先插入几个文档，它们的 JSON 像这样：

```
{
  "i": value
}
```

使用一个循环插入数据：

```
for(int i = 0; i < 100; i++) {
  coll.insert(new BasicDBObject().append("i", i));
}
```

我们注意到，同一个 coll，我们完全可以插入不同风格的数据，这就是 MongoDB 的重要特性“模式自由”。

十、统计文档数

现在我们已经有101份文档在数据库中了，现在统计一下看是否正确。

```
long count = coll.getCount();
System.out.println(count);
```

控制台将会输出：101

十一、使用游标取得所有的文档

```
DBCursor cursor = coll.find();

while(cursor.hasNext()) {
  DBObject object = cursor.next();
  System.out.println(object);
}
```

十二、查询单个文档

```
DBObject query = new BasicDBObject();

query.put("i", 71);

cursor = coll.find(query);
```

```
while(cur.hasNext()) {
  DBObject object = cursor.next();
  System.out.println(object);
}
```

控制台的输出类似如下：

```
{ "_id" : "49903677516250c1008d624e" , "i" : 71 , "_ns" : "testCollection"}
```

十三、查询文档集合

根据查询条件，我们可以通过 DBCollection 从数据库中取出多个对象，比如查询 i>50的文档集合：

```
query = new BasicDBObject();

query.put("i", new BasicDBObject("$gt", 50)); // i>50

cursor = coll.find(query);

while(cursor.hasNext()) {
  DBObject object = cursor.next();
  System.out.println(object);
}
```

比如查询条件为 20<i<=30：

```
query = new BasicDBObject();

// 20<i<=30
query.put("i", new BasicDBObject("$gt", 20).append("$lte", 30));

cursor = coll.find(query);

while(cursor.hasNext()) {
  DBObject object = cursor.next();
  System.out.println(object);
}
```

十四、创建索引

MongoDB 支持索引，并且给一个 DBCollection 添加索引非常简单，你只要指明需要创建索引的字段，然后指明其是升序(1)还是降序(-1)即可，比如在"i"上创建升序索引。

```
coll.createIndex(new BasicDBObject("i", 1)); // 1代表升序
```

十五、查询索引

我们可以查询到所有的索引：

```
List<DBObject> list = coll.getIndexInfo();

for(DBObject index : list)｛
  System.out.println(index);
}
```

控制台的输出类似如下所示：

```
{ "name" : "i_1" , "ns" : "mydb.testCollection" , "key" : { "i" : 1} , "_ns" : "system.indexes"}
```

MongoDB 的管理功能

一、获取所有的数据库

```
Mongo m = new Mongo();

for(String s : m.getDatabaseNames()) {
  System.out.println(s);
}
```

二、删除数据库

```
m.dropDatabase("my_new_db");
```

MongoDB 的 Java 类型

一、对象 ID

ObjectId 被用作自动生成的唯一 ID.

```
ObjectId id = new ObjectId();
ObjectId copy = new ObjectId(id);
```

二、正则表达式

```
Pattern john = Pattern.compile("joh?n", CASE_INSENSITIVE);
DBObject query = new BasicDBObject("name", john);

// 查询所有 "name" 匹配 /joh?n/i 的文档
DBCursor cursor = collection.find(query);
```

三、日期和时间

```
Date now = new Date();
DBObject time = new BasicDBObject("ts", now);

collection.save(time);
```

四、数据库引用

DBRef 可以用来保存数据库引用。

```
DBRef addressRef = new DBRef(db, "foo.bar", address_id);
DBObject address = addressRef.fetch();

DBObject person = BasicDBObjectBuilder.start()
  .add("name", "Fred")
  .add("address", addressRef)
  .get();
collection.save(person);

DBObject fred = collection.findOne();
DBRef addressObj = (DBRef)fred.get("address");
addressObj.fetch();
```

五、二进制数据

字节数组(byte[])被当作二进制数据。

六、内嵌文档

JSON 样式的数据如下：

```
{
  "x": {
    "y": 3
  }
}
```

则在 MongoDB 中，Java 表示为：

```
DBObject y = new BasicDBObject("y", 3);
DBObject x = new BasicDBObject("x", y);
```

七、数组

任何继承自 List 的对象，在 MongoDB 中，都被当成是数组。

如果想表示如下 JSON 数据：

```
{
  "x": [
    1,
    2,
    {"foo": "bar"},
    4
  ]
}
```

则在 Java 中，应该为：

```
List<Object> x = new ArrayList<Object>();
x.add(1);
x.add(2);
x.add(new BasicDBObject("foo", "bar"));
x.add(4);

DBObject doc = new BasicDBObject("x", x);
System.out.println(doc);
```