## update() 方法

update() 方法用于更新已存在的文档。语法格式如下：

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

})

**参数说明：**

* **query**: update的查询条件，类似sql update查询内where后面的。
* **update**: update的对象和一些更新的操作符（如$,$inc...）等，也可以理解为sql update查询内set后面的
* **upsert**: 可选，这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。
* **multi**: 可选，mongodb 默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。
* **writeConcern**:可选，抛出异常的级别。

### **实例**

我们在集合 col 中插入如下数据：

>db.col.insert({

title: 'MongoDB 教程',

description: 'MongoDB 是一个 Nosql 数据库',

by: '菜鸟教程',

url: 'http://www.runoob.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100})

接着我们通过 update() 方法来更新标题(title):

>db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) # 输出信息

> db.col.find().pretty()

{

"\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "菜鸟教程",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"database",

"NoSQL"

],

"likes" : 100}

>

以上语句只会修改第一条发现的文档，如果你要修改多条相同的文档，则需要设置 multi 参数为 true。

>db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}},{multi:true})

## save() 方法

save() 方法通过传入的文档来替换已有文档。语法格式如下：

db.collection.save(

<document>,

{

writeConcern: <document>

})

**参数说明：**

* **document**: 文档数据。
* **writeConcern**:可选，抛出异常的级别。

### **实例**

以下实例中我们替换了 \_id 为 56064f89ade2f21f36b03136 的文档数据：

>db.col.save({

    "\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "Runoob",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"NoSQL"

],

"likes" : 110})

替换成功后，我们可以通过 find() 命令来查看替换后的数据

>db.col.find().pretty(){

"\_id" : ObjectId("56064f89ade2f21f36b03136"),

"title" : "MongoDB",

"description" : "MongoDB 是一个 Nosql 数据库",

"by" : "Runoob",

"url" : "http://www.runoob.com",

"tags" : [

"mongodb",

"NoSQL"

],

"likes" : 110}

>