mongodb

[1 概述 4](#_Toc466481712)

[1.1 概述，mongodb是文档型非关系数据库 4](#_Toc466481713)

[1.2 文档型存储系统 4](#_Toc466481714)

[2 安装 6](#_Toc466481715)

[2.1 linux 6](#_Toc466481716)

[2.2 windows 6](#_Toc466481717)

[3 管理mongodb 9](#_Toc466481718)

[3.1 C/S架构 9](#_Toc466481719)

[3.2 服务器端：mongod 9](#_Toc466481720)

[3.2.1 linux 9](#_Toc466481721)

[3.2.2 windows 9](#_Toc466481722)

[3.3 客户端：mongo 12](#_Toc466481723)

[4 操作mongodb的核心概念 14](#_Toc466481724)

[4.1 文档型数据库系统 14](#_Toc466481725)

[4.2 文档-集合-数据库，三级层级组成结构 14](#_Toc466481726)

[4.3 文档是独立 14](#_Toc466481727)

[4.4 BSON，JSON 15](#_Toc466481728)

[5 数据库操作 17](#_Toc466481729)

[5.1 show dbs 17](#_Toc466481730)

[5.2 db 17](#_Toc466481731)

[5.3 选择默认库，use db-name 17](#_Toc466481732)

[5.4 删除库，db.dropDatabase() 18](#_Toc466481733)

[5.5 查看库状态，db.stats() 18](#_Toc466481734)

[5.6 获取库级别帮助，db.help() 18](#_Toc466481735)

[6 集合操作 19](#_Toc466481736)

[6.1 查看所有集合，show collections 19](#_Toc466481737)

[6.2 创建集合，db.createCollection('collection-name') 19](#_Toc466481738)

[6.3 删除集合，db.collection-name.drop() 19](#_Toc466481739)

[6.4 改名集合，db.collection-name.renameCollection('new-collection-name') 20](#_Toc466481740)

[6.5 获取集合帮助，db.collection-name.help() 20](#_Toc466481741)

[6.6 获取集合状态，db.collection-name.stats() 20](#_Toc466481742)

[7 文档操作 21](#_Toc466481743)

[7.1 CRUD 21](#_Toc466481744)

[7.1.1 增，db.collection-name.insert() 21](#_Toc466481745)

[7.1.2 查，db.collection-name.find() 21](#_Toc466481746)

[7.1.3 改，db.collection-name.update() 21](#_Toc466481747)

[7.1.4 删，db.collection-name.remove() 22](#_Toc466481748)

[7.2 ObjectID 22](#_Toc466481749)

[7.3 基本语法 23](#_Toc466481750)

[7.4 条件的处理 24](#_Toc466481751)

[7.4.1 概述 24](#_Toc466481752)

[7.4.2 关系运算符 24](#_Toc466481753)

[7.4.3 逻辑运算符 26](#_Toc466481754)

[7.4.4 正则检测 27](#_Toc466481755)

[7.4.5 数组元素检测 28](#_Toc466481756)

[7.4.6 对象的比较。（内联文档） 30](#_Toc466481757)

[7.4.7 $where,自定义条件 $where: function() {return true|false} 31](#_Toc466481758)

[7.5 查询操作 find() 32](#_Toc466481759)

[7.5.1 查询条件 32](#_Toc466481760)

[7.5.2 属性过滤 32](#_Toc466481761)

[7.5.3 分组-统计，group() 33](#_Toc466481762)

[7.5.4 排序结果,find().sort(); 34](#_Toc466481763)

[7.5.5 截取文档find().skip() find().limit() 35](#_Toc466481764)

[7.5.6 统计文档数collection.count() 36](#_Toc466481765)

[7.5.7 去重统计collection.distinct() 36](#_Toc466481766)

[7.5.8 游标，cursor 37](#_Toc466481767)

[7.6 更新 39](#_Toc466481768)

[7.6.1 常规更新 39](#_Toc466481769)

[7.6.2 upsert，更新插入 39](#_Toc466481770)

[7.6.3 属性更新，修改器 40](#_Toc466481771)

[7.6.4 更新多条 42](#_Toc466481772)

[8 索引 44](#_Toc466481773)

[8.1 索引类型 44](#_Toc466481774)

[8.1.1 普通索引，对关键字没有要求 44](#_Toc466481775)

[8.1.2 唯一索引，要求关键字唯一 44](#_Toc466481776)

[8.1.3 2d平面空间索引，要求关键字是，坐标。x,y 44](#_Toc466481777)

[8.2 管理索引 44](#_Toc466481778)

[8.2.1 添加索引 44](#_Toc466481779)

[8.2.2 查看索引 45](#_Toc466481780)

[8.2.3 删除索引 45](#_Toc466481781)

[8.2.4 选项：指定索引名字 46](#_Toc466481782)

[8.2.5 选项：唯一索引 46](#_Toc466481783)

[8.3 平面空间2d索引 47](#_Toc466481784)

[9 功能概要 50](#_Toc466481785)

[9.1 复制，replicate 50](#_Toc466481786)

[9.2 分区（分片） 50](#_Toc466481787)

[9.3 map/reduce 50](#_Toc466481788)

[10 PHP操作mongodb 51](#_Toc466481789)

[10.1 加载操作mongodb的扩展 51](#_Toc466481790)

[10.2 基本语法 52](#_Toc466481791)

[10.3 获取集合中的全部数据 54](#_Toc466481792)

[10.4 增加条件获取 54](#_Toc466481793)

[10.5 过滤字段 54](#_Toc466481794)

[10.6 排序查询sort 55](#_Toc466481795)

[10.7 skip()，limit() 55](#_Toc466481796)

[10.8 分组查询,group() 55](#_Toc466481797)

[10.9 增 56](#_Toc466481798)

[10.10 删除 56](#_Toc466481799)

[10.11 修改 56](#_Toc466481800)

[11 mongodb的使用 57](#_Toc466481801)

[11.1 哪里用?为什么用？ 57](#_Toc466481802)

[11.2 不能用在哪里？ 57](#_Toc466481803)

[11.3 数据结构的设计 57](#_Toc466481804)

# 概述

mongodb：是介于关系型与非关系之间的一种数据库系统！

## 概述，mongodb是文档型非关系数据库

是一种数据库，类似MySQL。

不同于MySQL的是：是一种“非关系型数据库”。

非关系型数据库：NOSQL-(Not Only SQL, non-relation)。不以关系型（二维表）进行数据存储结构的数据库的统称。包括：memcached（内存型数据库），文件系统，redis。Mongodb。

发展到今天：典型的非关系有：

* key-value型。
* 文档型（document）（mongodb的存储结构类型）
* 图型（大数据）

## 文档型存储系统

类似于，关系型（二维表）。

二维表中的记录 映射成 文档型的文档。

区别是：文章 和 文章分类

二维表：

article表

|  |  |  |
| --- | --- | --- |
| Article\_id |  |  |
| Article\_content |  |  |
| Category\_id | 所属分类ID | 关联字段 |

category表

|  |  |  |
| --- | --- | --- |
| Category\_id |  |  |
| Category\_title |  |  |

文档型：

将当前内容相关的全部的数据，存储为一个文档。

站在文章的角度去存储。会得到下面的文档：

内容：北京PHP训练营，开营了！

分类：PHP培训

”理解成：没有满足范式设计的数据表“。

优势：存储，提取。不需要做大量的关联操作，就可以获相关的全部属性。

劣势：更新，维护。出现大量的数据冗余。

可见：

项目中：

日志，文章管理部分，存储到mongodb。会得到较高的存取速度。

复杂，核心数据的维护，存储关系型数据。

# 安装

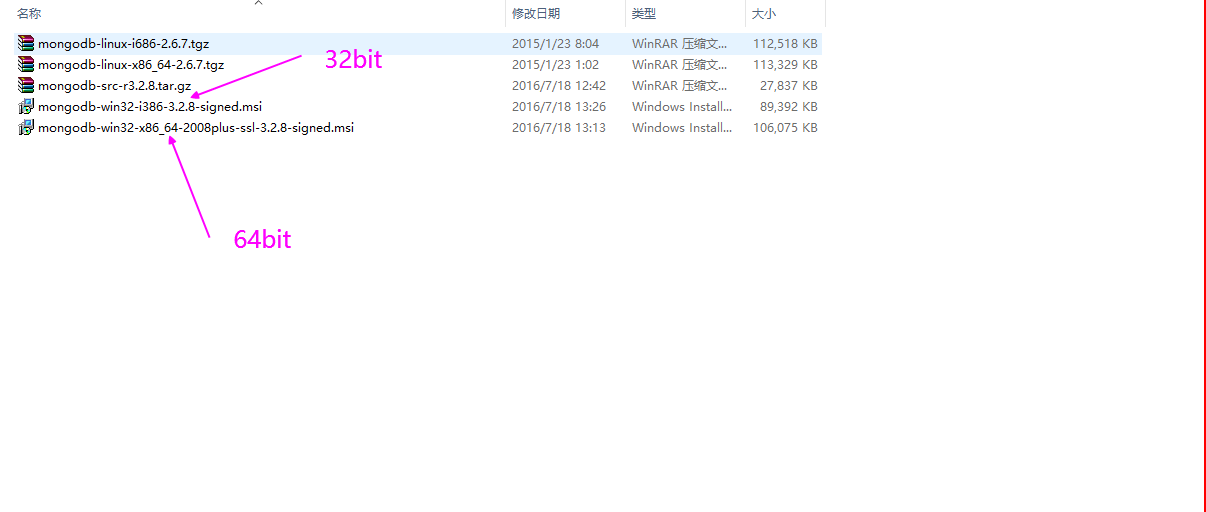
## linux

## windows

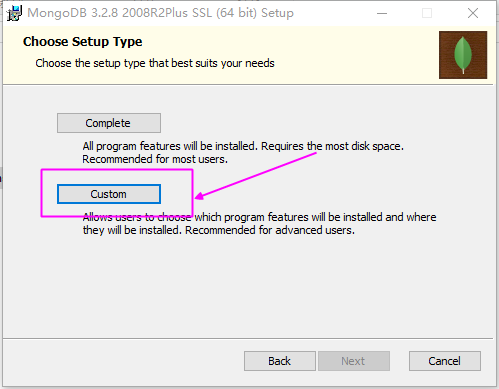
下载：获取最新的mongodb

<https://www.mongodb.com/download-center>

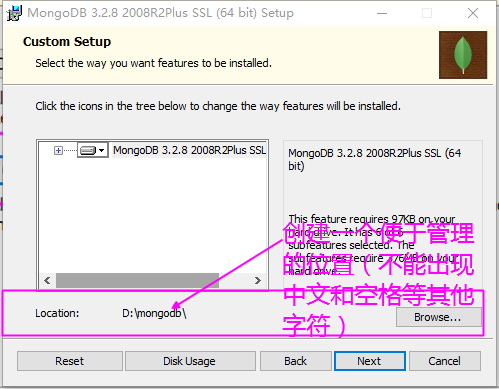
3.2.x 目前最新的版本。



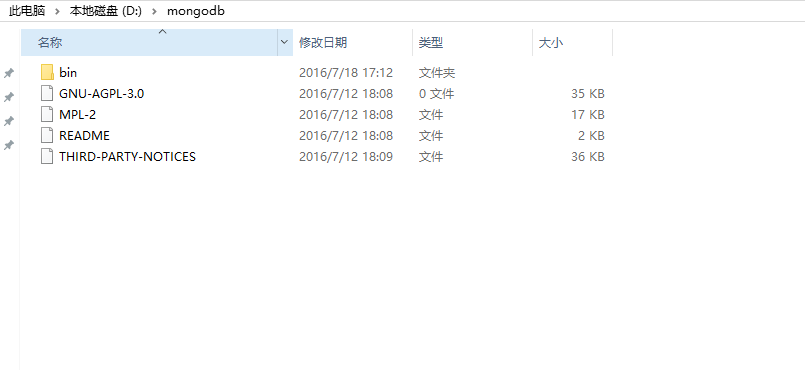
运行程序安装即可



配置安装路径



成功结果：



# 管理mongodb

## C/S架构

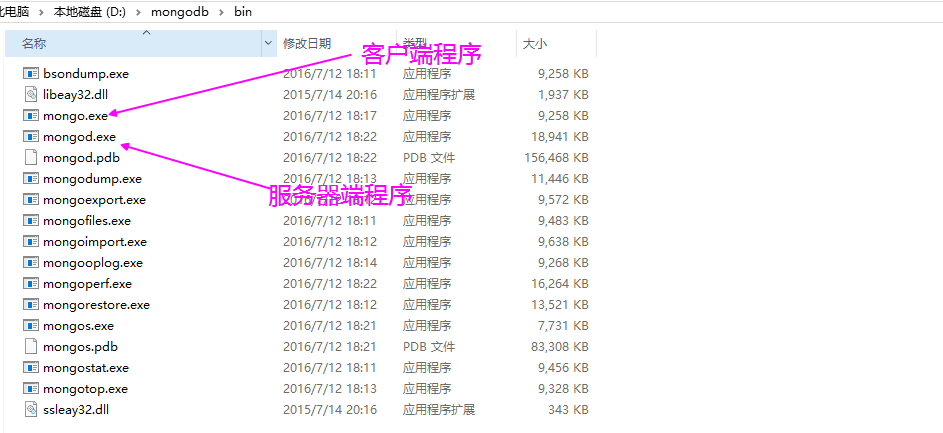
## 服务器端：mongod

### linux

### windows

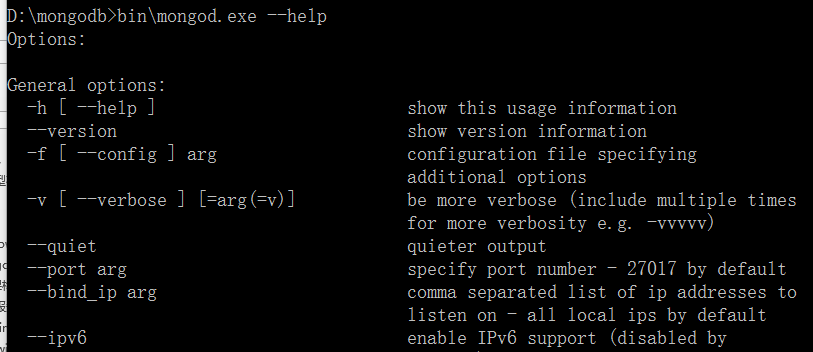
daemon

bin/mongod



#### CMD

bin/mongod –help



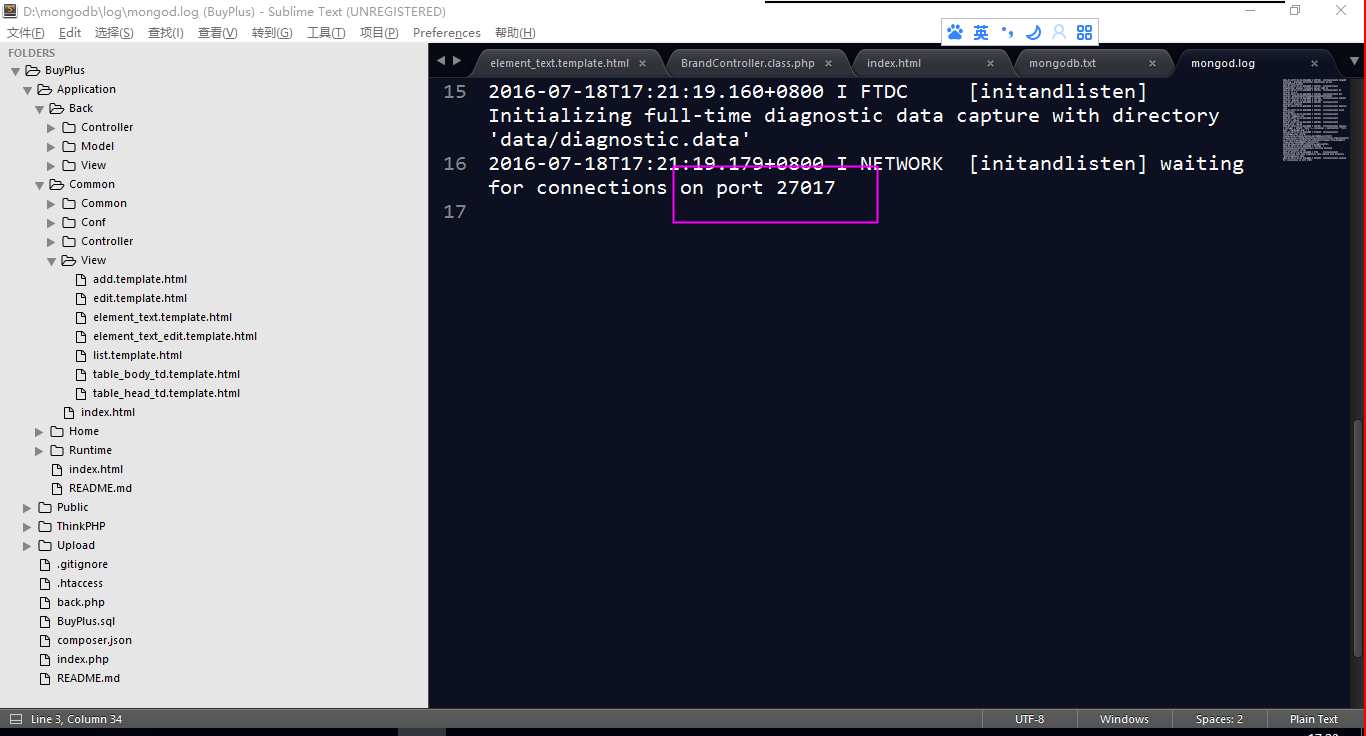
配置：数据目录。（必要）

配置：日志文件。（可选，强烈建议）

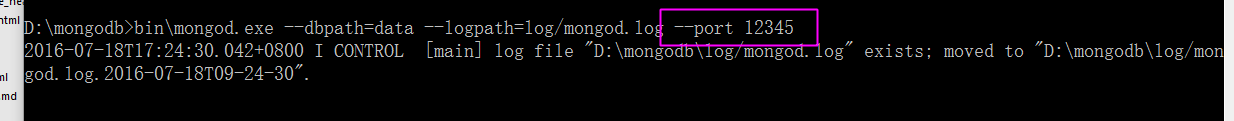


通过运行日志可知：默认端口是27017

log/mongod.log



选项 --port 自定义端口，来设置新的监听端口



如果是32bit的同学：

启动时，增加选项： --storageEngine=”MMAPv1” 表示选择mmapv1 存储引擎的意思。

#### 服务形式管理

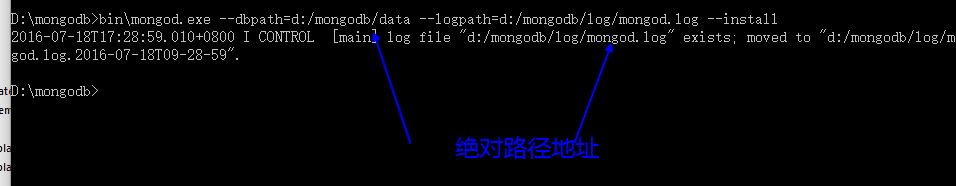
安装成windows的服务。

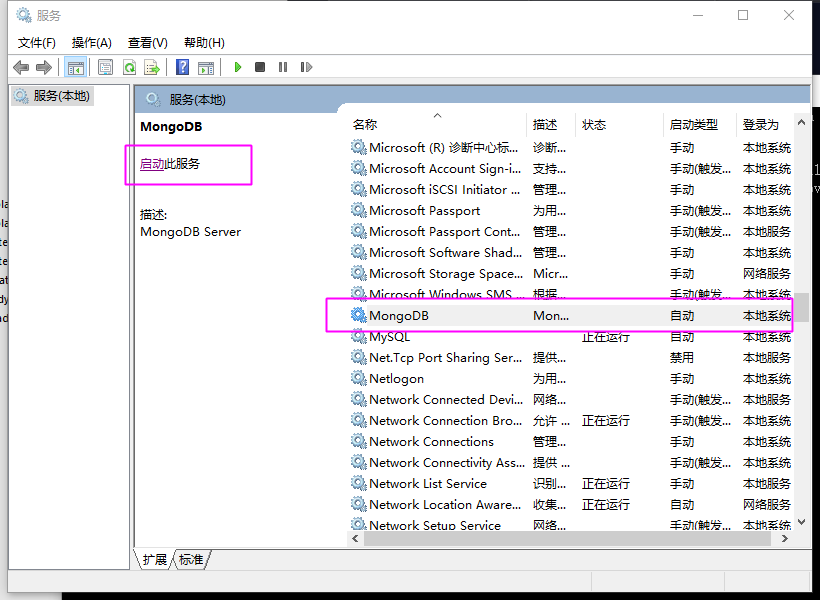
mongod --install

mongod --remove

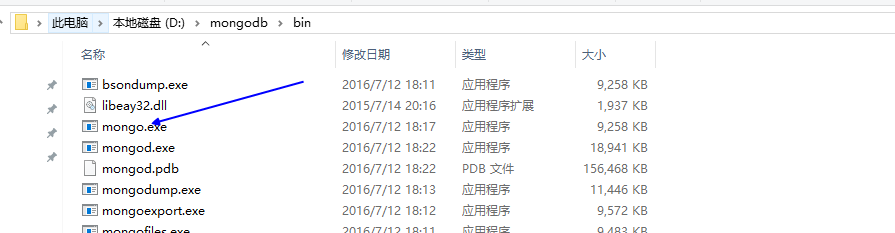
mongod --reinstall

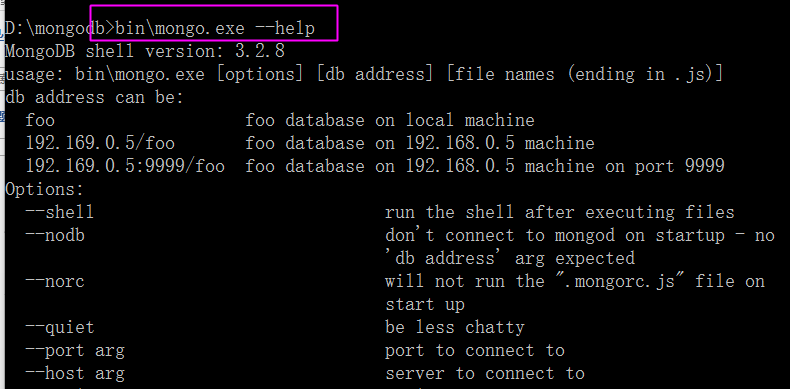
安装：



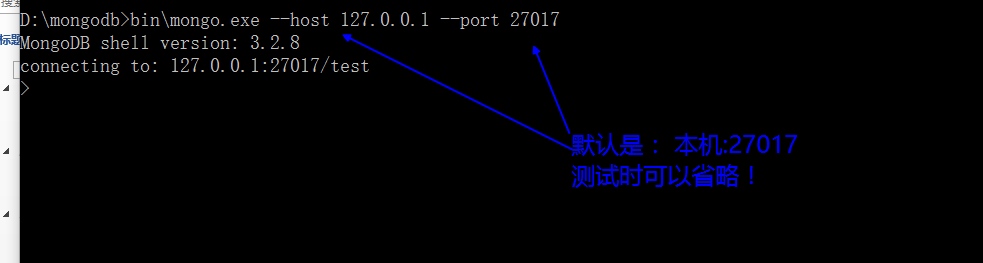


## 客户端：mongo





--host, --port 指定连接目标服务器地址



# 操作mongodb的核心概念

## 文档型数据库系统

参考：1.2章节

存储结构

## 文档-集合-数据库，三级层级组成结构

1. 数据库，database
2. 集合，collection
3. 文档，document

文档存储与集合中，集合存储与数据库中。

对比关系型数据库系统：

记录存储与表中，表存储与数据库中。



## 文档是独立

同一个集合中的文档的结构，是可以不相同的。而没有逻辑关系的。是独立的。

例如，同一个集合 Student中的文档：



理解：

文档就是定义在不同的命名空间的数据变量。

文档：变量。（复杂的对象类型）

命名空间：由 数据库名.集合名 构成的。

## BSON，JSON

BSON： Binary JSON，可执行的JSON。

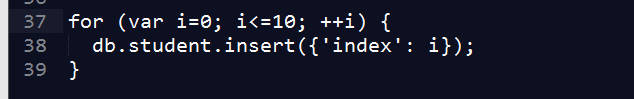
mongo的客户端语法，基于JSON的可执行版本。

体现：

文档：就是JSON格式。

mongo客户端程序：JSON（BSON）的解释器（类似于浏览器解释JavaScript）。

例如：执行一段循环：可以循环插入11条记录。



JS全栈：

浏览器端：

移动端：

移动端浏览器

移动端APP：

native原生。

Web APP（H5+CSS3+JS）。网页被套壳。

Hybrid APP：混合APP

情况一（实现混合）：复杂功能由原生实现，展示功能由Web APP实现。

情况二（开发混合）：使用Html+CSS+JS开发，转换为 原生程序。（react native for iso）

服务器端

web服务器：nodeJS

数据库服务器：mongodb

业务逻辑语言：javaScript

语言 和 运行环境（运行虚拟机）

# 数据库操作

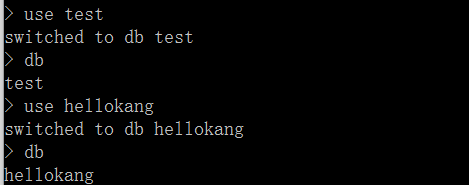
## show dbs

查看所有库



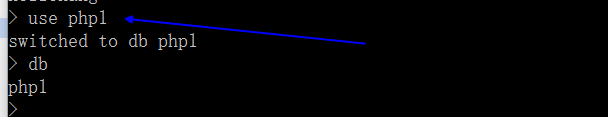
## db

当前库对象



## 选择默认库，use db-name

选择默认库

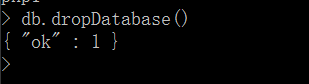


如果不在库中增加集合或文档，则库还会消失。

如果创建了集合文档，库就会被自动创建。

## 删除库，db.dropDatabase()

删除当前的库。



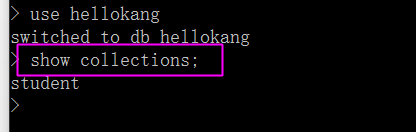
## 查看库状态，db.stats()

## 获取库级别帮助，db.help()

# 集合操作

## 查看所有集合，show collections

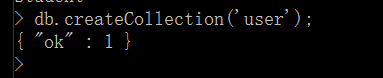
当前库下的集合列表



## 创建集合，db.createCollection('collection-name')

库创建集合，

当前库对象.createCollection(‘集合名‘)

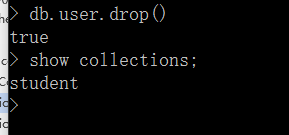


## 删除集合，db.collection-name.drop()

一但创建了集合，在当前数据库对象上，增加一个集合名属性。引用该的集合。

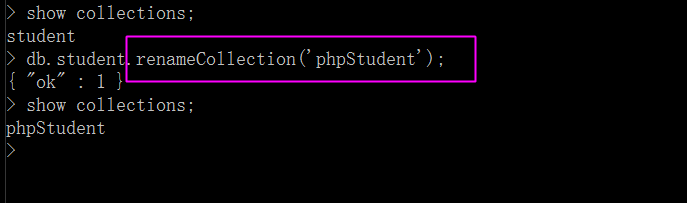
通过该集合对象的方法，就可以操作集合。

删除集合



## 改名集合，db.collection-name.renameCollection('new-collection-name')

改名



## 获取集合帮助，db.collection-name.help()

## 获取集合状态，db.collection-name.stats()

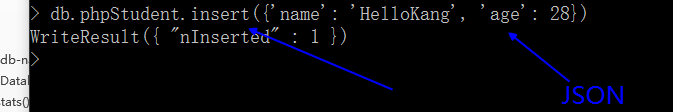
# 文档操作

## CRUD

### 增，db.collection-name.insert()

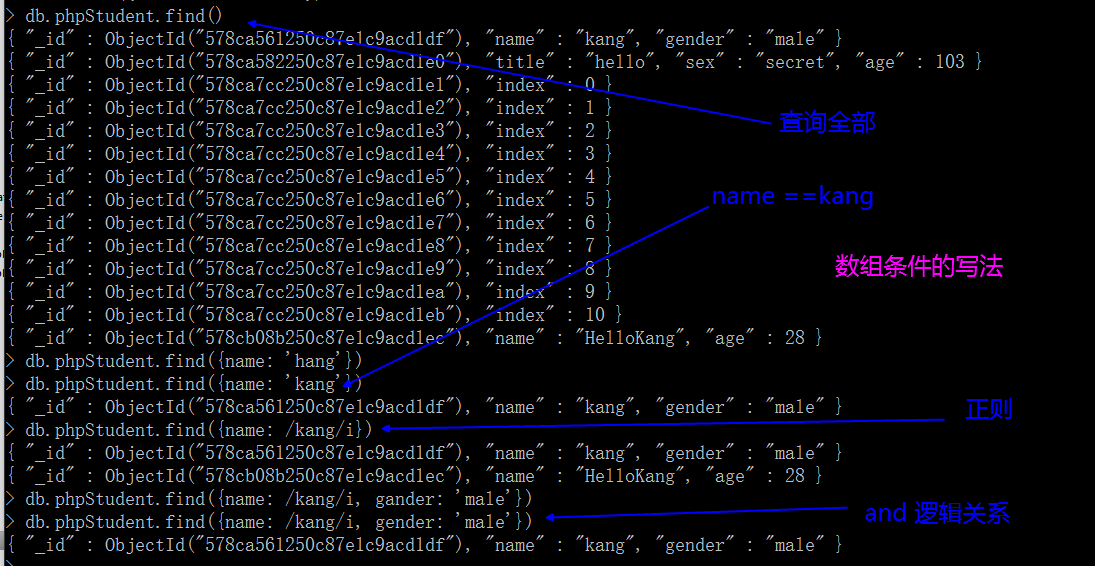
db.collection-name.insert(json对象)

每次插入一个文档对象。



### 查，db.collection-name.find()

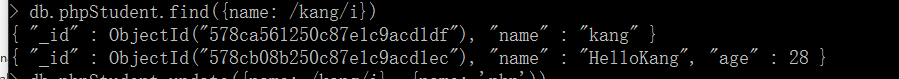
db.collection-name.find([条件对象, 属性过滤对象])



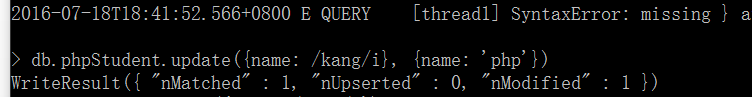
### 改，db.collection-name.update()

db.collection-name.update(条件，更新内容)

当前数据

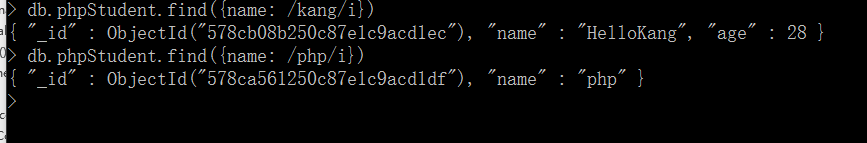


修改：



结果：

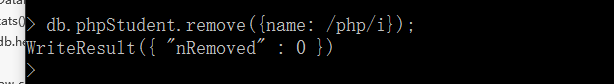
默认修改一条



可以通过 第四个参数进行修改：

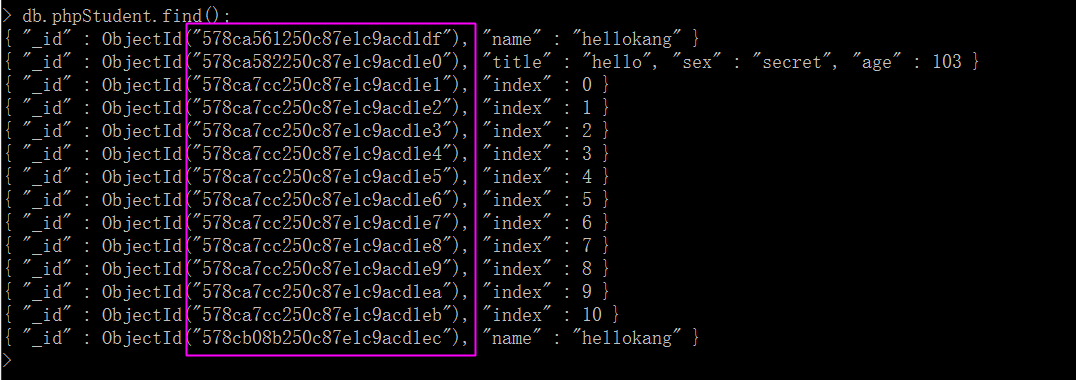
### 删，db.collection-name.remove()

db.collection-name.remove({条件})



## ObjectID

文档的内部标识。mongodb内部使用的。非业务逻辑数据。



了解：

生成原则：时间，机器名要素生成。

578ca7cc250c87e1c9acd1e2

时间戳（秒）

主机标识

进程标识

顺序号

## 基本语法

* 大小写敏感。
* 类型敏感：6 与 ‘6’ 是不同的数据，强类型
* 有序键值对：json键值对。{name: ‘kang’, age: 30} 与 {age:30, name:’kang’}不是同一个数据。
* 面向对象编程。
* 与JavaScript语法保持一致

## 条件的处理

### 概述

查询.find(query)，删除.remove(query)，修改.update(query)。

query，就是查询条件。

通用的语法：

json对象。

每个属性表示一个条件。（或其他运算）

通用的格式：

{属性: {关系运算符: ‘比较值’}}

tip：参考 框架中 关于查询条件的数组写法：[‘age’=>[‘gt’, 50]]

### 关系运算符

#### $lt, 小于

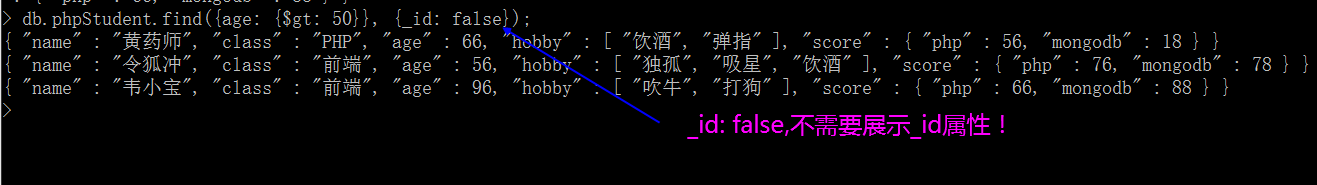
#### $lte，小于等于

#### $gt，大于

#### $gte，大于等于

#### $ne，不等于

例如：年龄大于50的学生



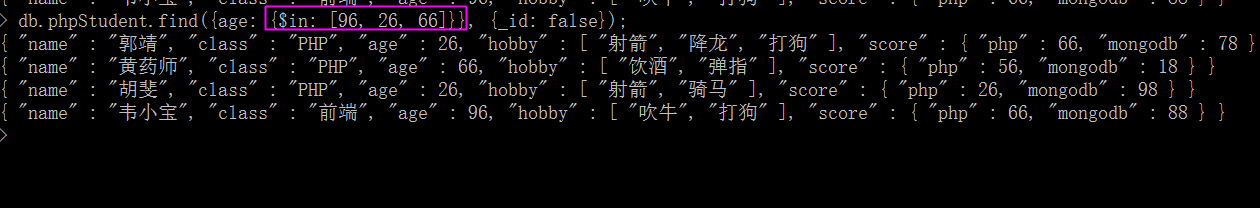
#### $in，在某个集合中

{'$in': [93, 96]}

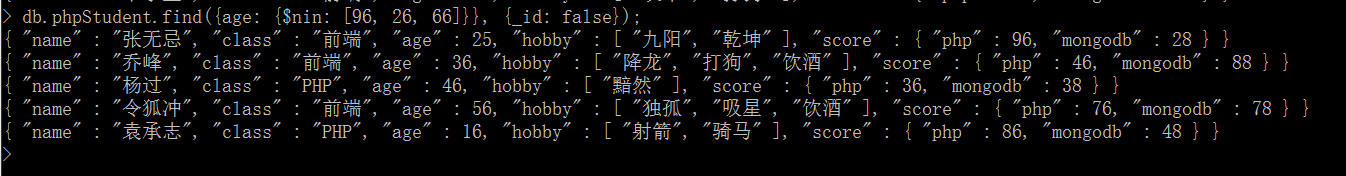
#### $nin，不在某个集合中

{'$nin': [93, 96]}

例如：年龄 为96， 66， 26 的人：



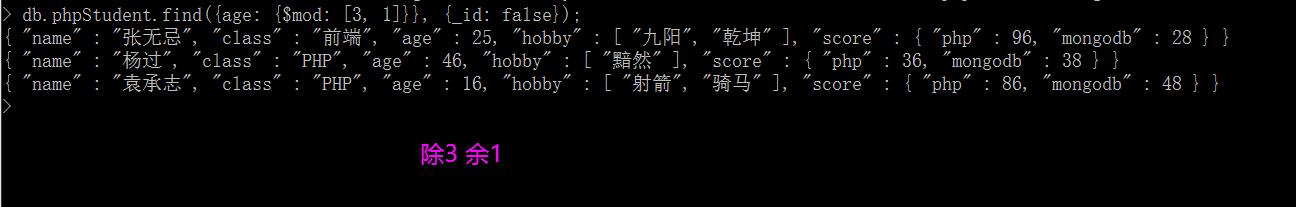
不在集合：



#### $mod，计算余数比较

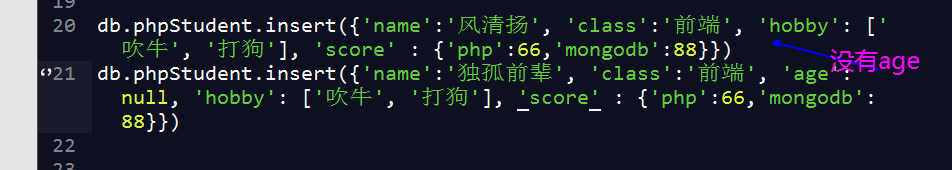
{'$mod': [3, 1]}，找到对3取余，余数为1的数据。

接受 数组型参数，第一个元素表示对几取余。第二个元素，表示余数为几。



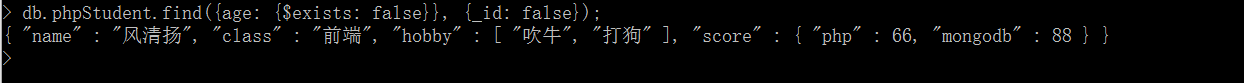
#### $exists，是否存在某个属性

{$exists:false}，某个属性是否存在。



检索，没有age属性的文档：

db.phpStudent.find({age: {$exists: false}}, {\_id: false});

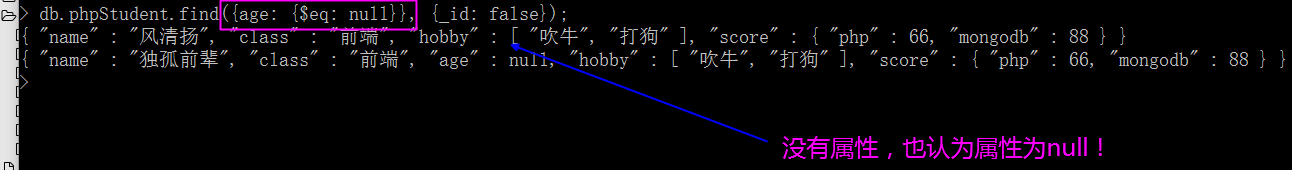


没检索到 age为null的文档。

与 null的比较：

如果属性为null，算存在！$exists:true, 不能匹配 属性为null的文档

检索 属性值为null：不存在属性的文档也被检索到。



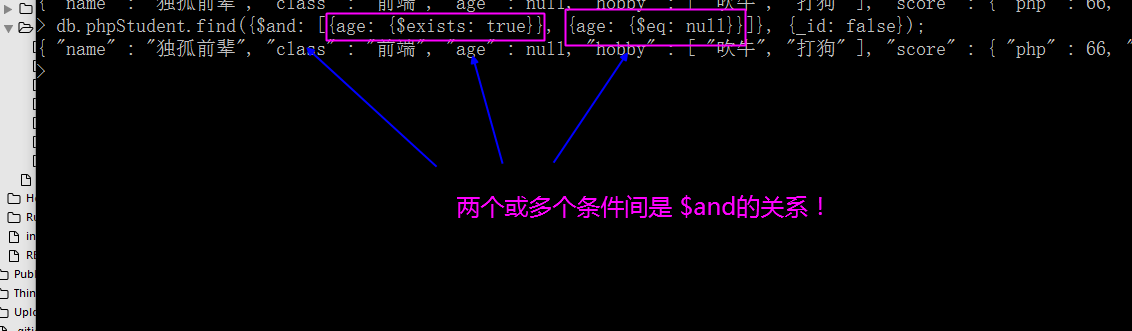
如果需要检测 存在，但是为null的属性文档呢？

$exists: true, $eq:null

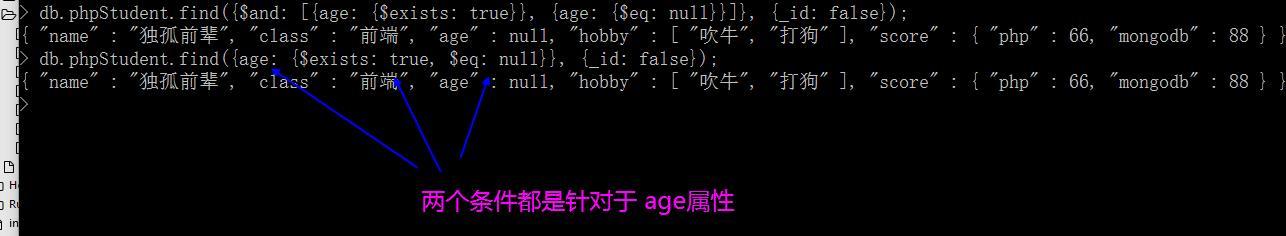
### 逻辑运算符

#### $and

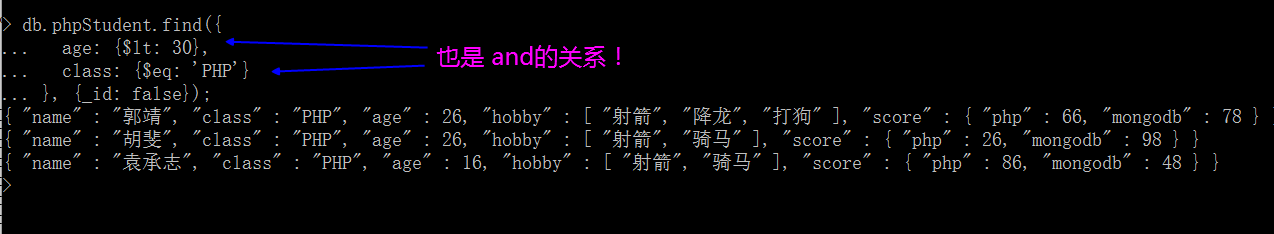
{$and: [条件1, 条件2]}，通用：



如果两个条件，是针对于同一个属性（字段），可以将$and的关系间写成：



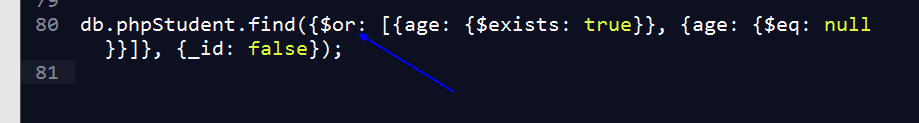
如果多个属性存在条件，条件间是 $and关系，可以间写成：

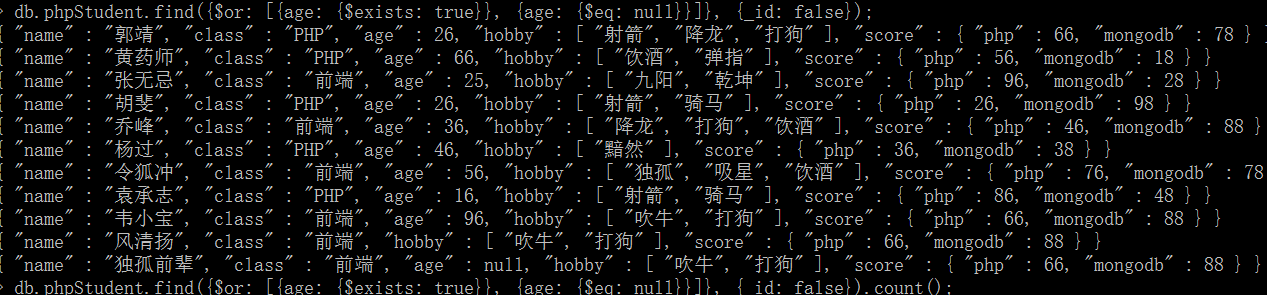


#### $or

{$or: [条件1， 条件2， ..]}

支持的通用语法：

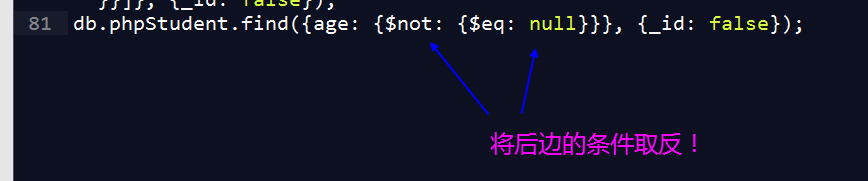




没有相应的简写语法！

#### $not

获取年不为null的数据：

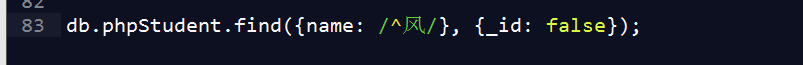


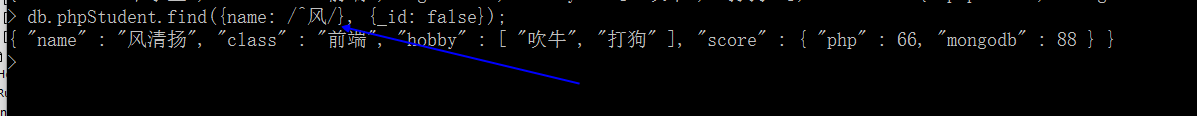
### 正则检测

{属性: 正则表达式对象}

//

例如：name以风开头：



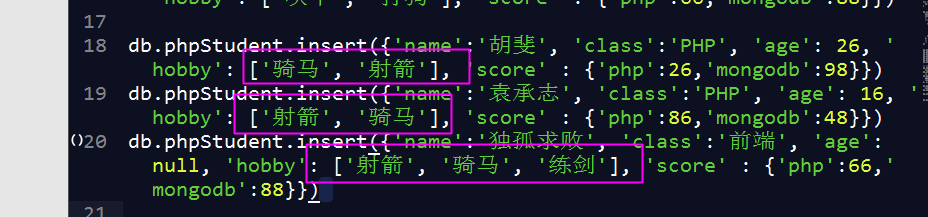


### 数组元素检测

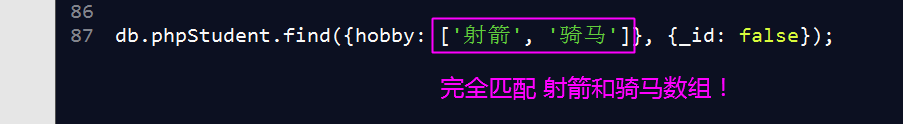
#### 完全匹配 :[]

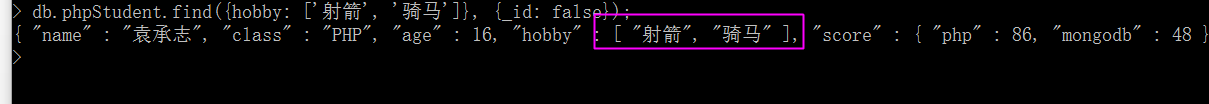
与给定的数组保持绝对一致：元素和顺序

测试数据：



检索

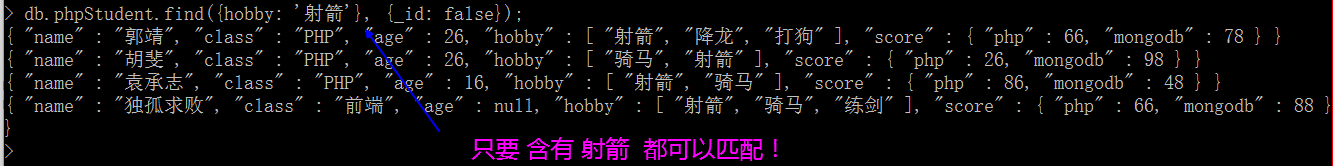




比较是有序的！

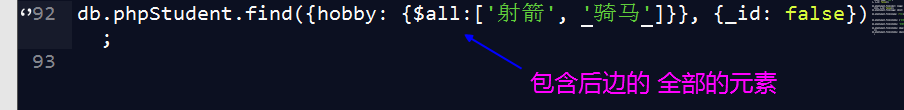
#### 包含匹配 : value



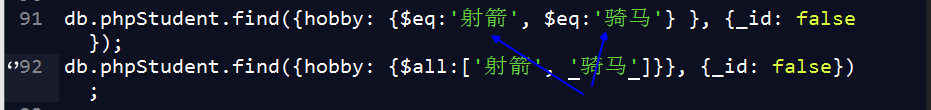


#### 包含全部 $all

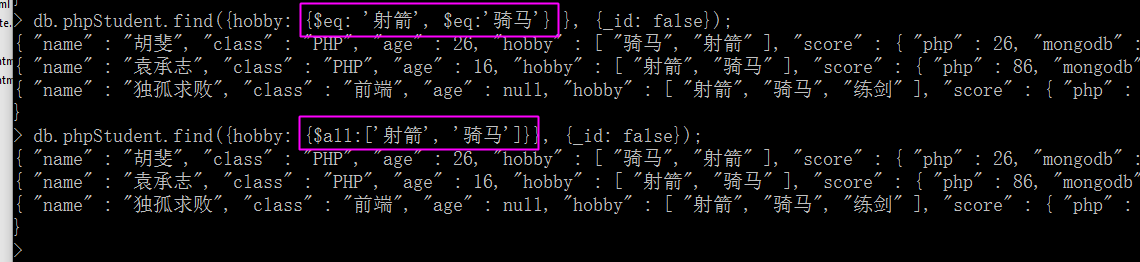
{$all:[元素列表]}



可以使用 and关系，改写：（不推荐，如果元素过多，语法会复杂）

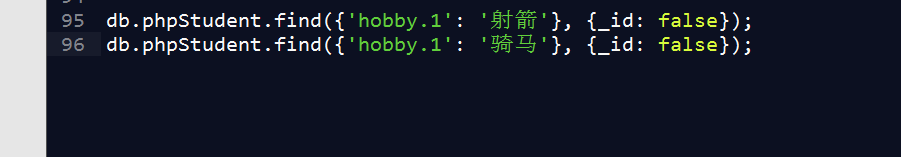


结果一致



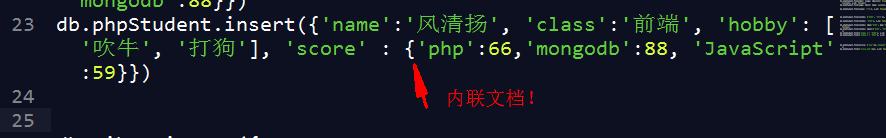
#### 数组索引对应匹配, property.index

‘属性.index’, 表示该索引值的元素



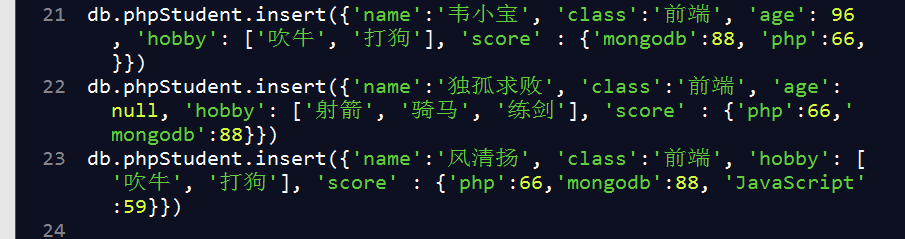


### 对象的比较。（内联文档）



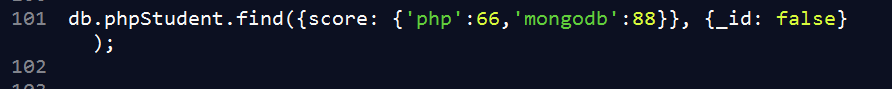
嵌入json对象内部的json对象，称之为内联文档！

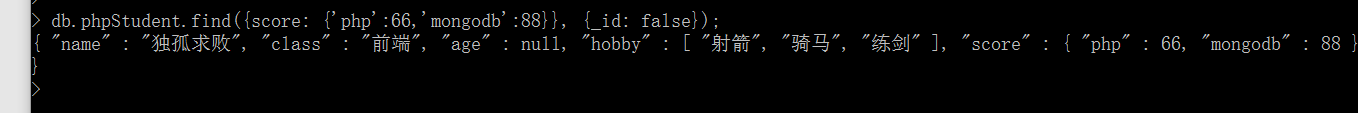
测试数据：



#### 完全匹配 : {}

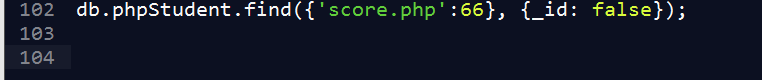
严格与给定的对象保持一致。顺序和键值对！

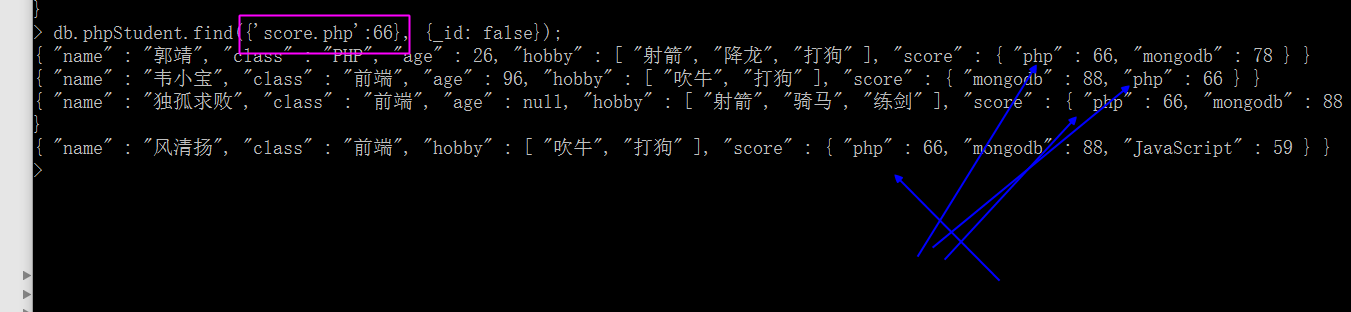




#### 对应的属性匹配 ‘propertyObject.property’:’value’

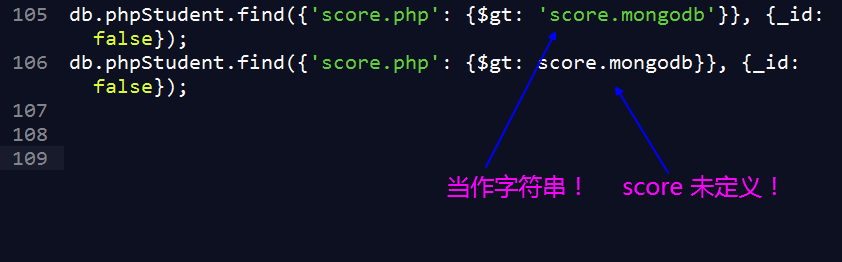
查找：php成绩为66 的：





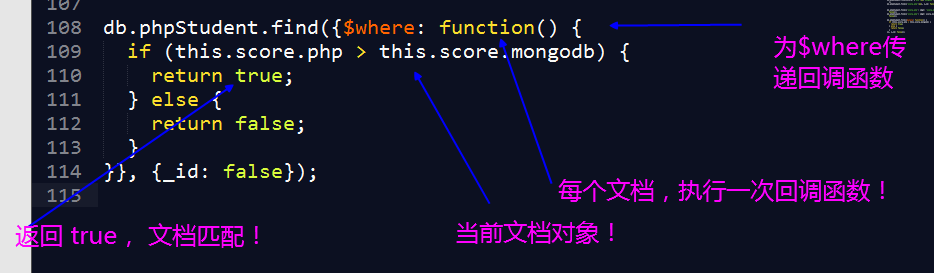
### $where,自定义条件 $where: function() {return true|false}

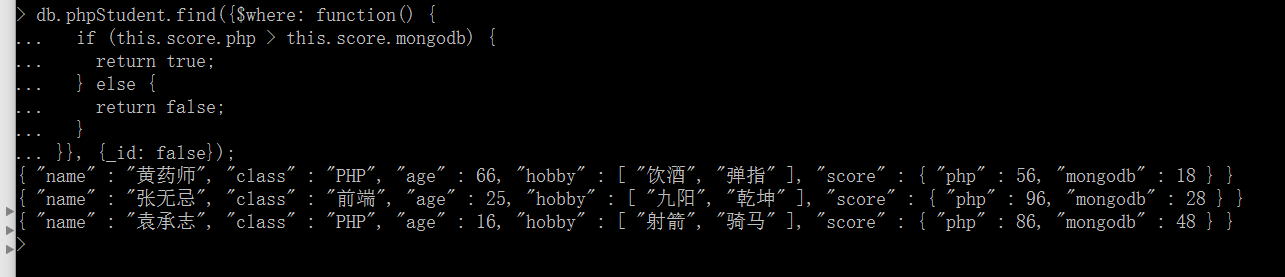
例如：PHP成绩 高于 mongodb 成绩的同学，错误的方法：



由于是 非关系，结构，不能做，属性间的横向比较！

使用自定义条件完成：





## 查询操作 find()

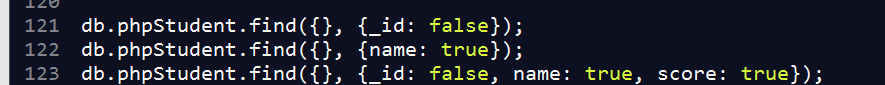
find(query, property)

### 查询条件

参考：7.4

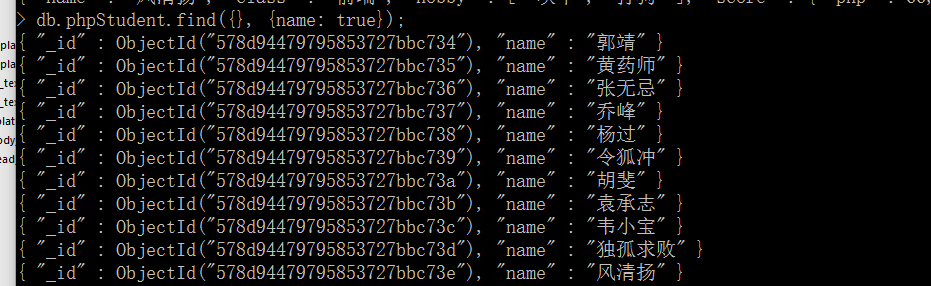
### 属性过滤

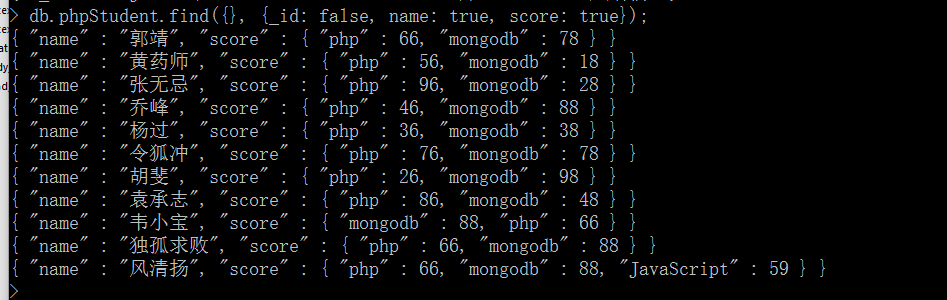
默认是全部属性。



如果独立某个属性为true，则其他属性不予展示。

\_id例外。





### 分组-统计，group()

group({

key: 分组属性,

condition: 分组前的过滤,

initial: 组内初始化数组,

$reduce: 文档回调

finalize: 组内结尾回调

})

逻辑如下：

依据 key 给定的属性进行分组。如果给了condition，使用对应的条件筛选掉不满足的文档。

initial: 每组对应的数据对象初始值。

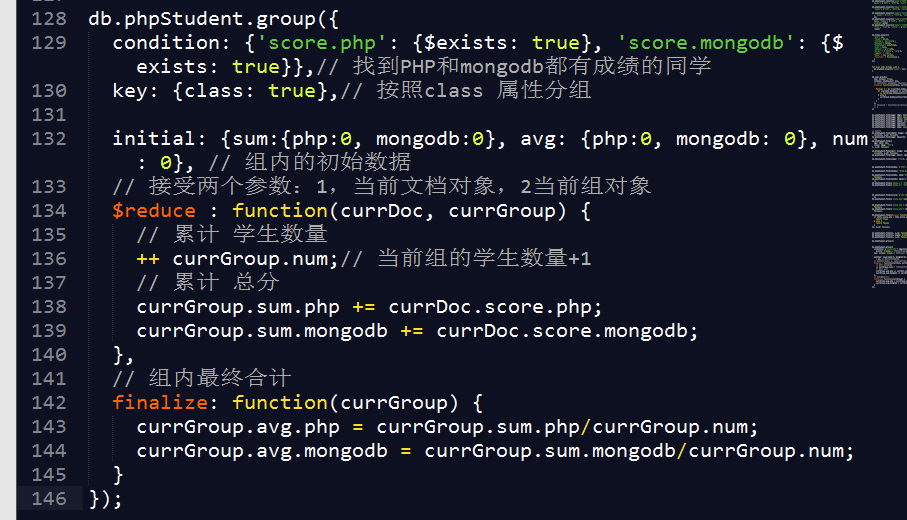
$reduce: 函数，针对于每个文档都会调用。用该函数完成对组内数据的统计处理。

自定义的组内统计函数。

finalize: 每组结束时，自动调用的回调函数

测试：例如：按照班级统计学生信息

统计学科 考试的平均成绩：获取总成绩，知道班级内的人数。



执行次数如图所示：



### 排序结果,find().sort();

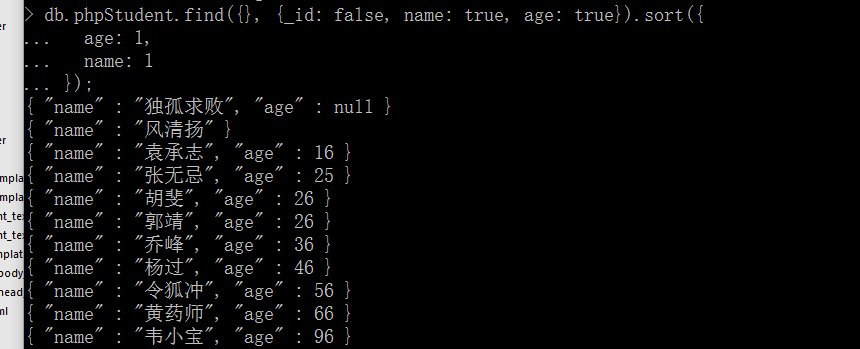
{排序属性: 1, 排序属性:-1}

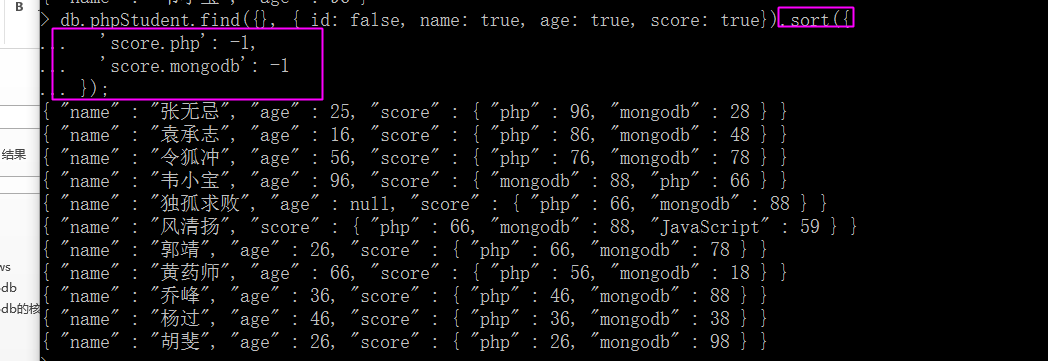
1. 升序

-1，降序

如果存在多个排序属性，先依据属性1排，再依据属性2排。

例如：



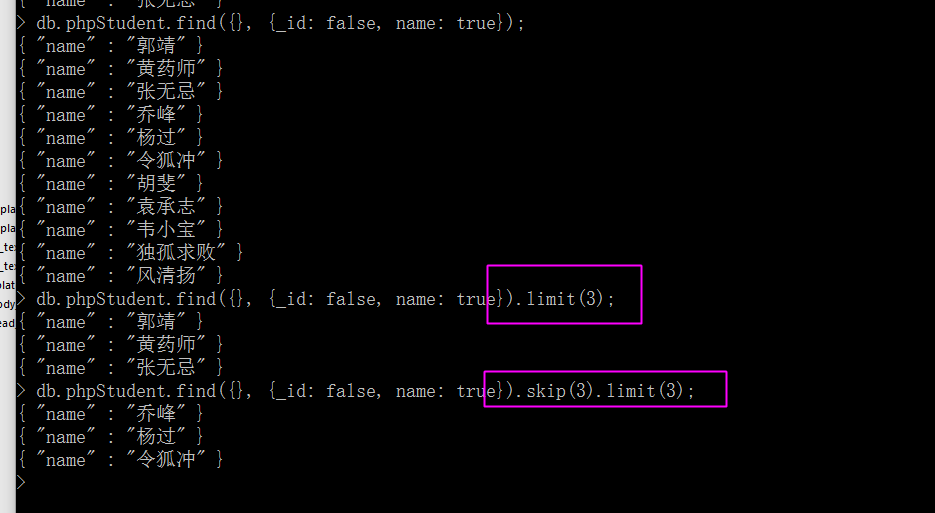


### 截取文档find().skip() find().limit()

find().skip(Number),跳过NuMber条记录

find().limit(Number)，表示仅仅获取前Number条记录

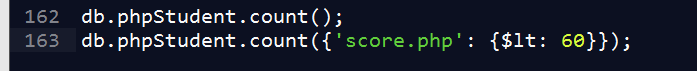
例如：

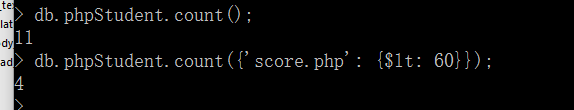


### 统计文档数collection.count()

count({条件})

条件可选，表示全部。

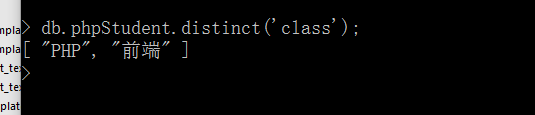




### 去重统计collection.distinct()

不是针对文档，而是针对于属性。

collection.distinct(属性名)



### 游标，cursor

db.collection.find() 方法返回的是游标。

var cursor = db.collection.find();

游标：

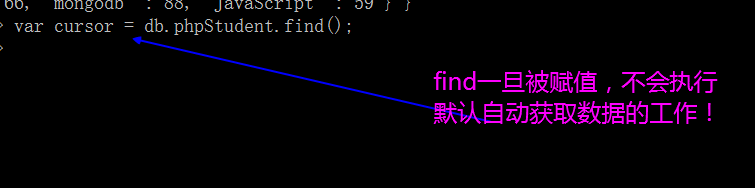
查询得到的文档结果集的引用。

当执行find()后，数据还没有被客户端所获取。而是将被查询的文档，使用游标所引用。

当执行find()而不去赋值时，mongo客户端，会自动的执行从游标获取数据。



相对：



相当于，去班级找10名会书法的同学，没有将10名同学叫过来，而仅仅记录其名字

仅仅得到了一份名单。

方法：sort(), limit(),skip(), 都是游标对象上的方法。操作的是游标，（是名单而已）

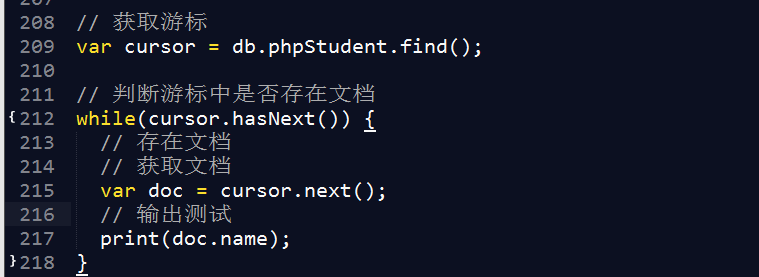
#### hasNext();

是否存在下一个。

#### next();

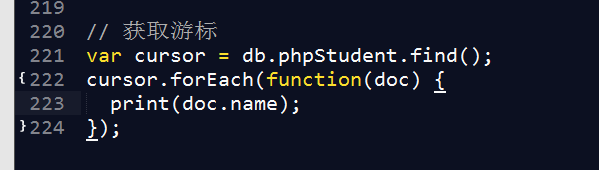
获取游标中的下一个数据。

配合 循环结构，获取游标中的全部文档。



#### cursor.forEach();建议推荐

迭代游标。

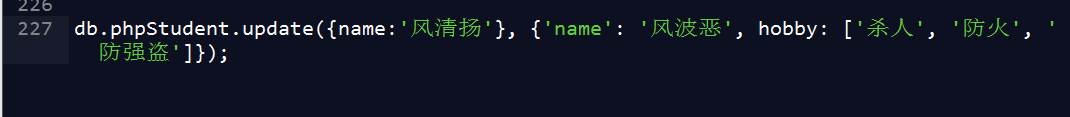


内部的优化措施：会一次性通过游标获取一定量的数据（不仅仅是一个文档），下次获取时，速度被优化。

## 更新

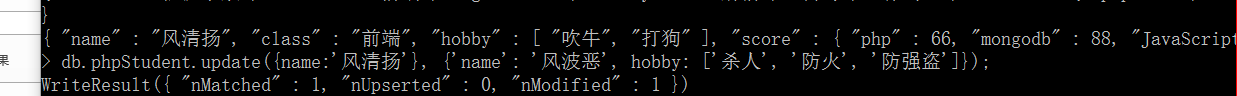
### 常规更新

db.collection-name.update(query, bson)

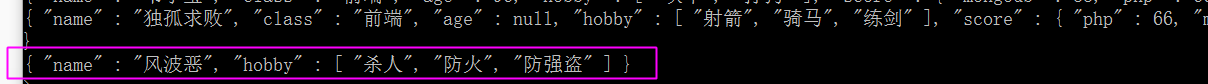


常规的模式，使用新文档，替换 旧文档，而不是进行 属性的更新set工作。

执行：



结果：



如果条件匹配多条，仅仅可以更新一条。

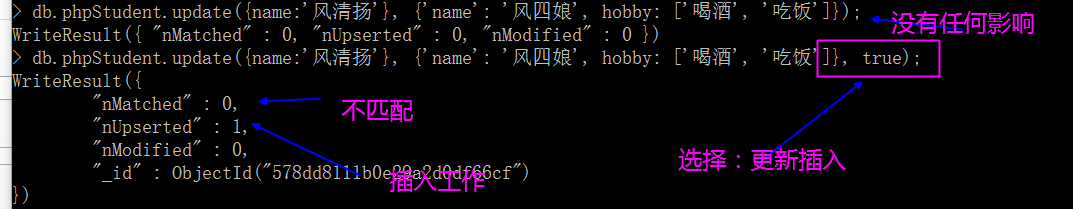
要求：使用具有标志性的条件，作为更新条件。

### upsert，更新插入

update - insert

如果条件匹配失败，则选择执行 插入操作。

默认的是没有该行为的，可以通过传递update方法的第三个参数，进行设置。true表示，执行upsert操作！



### 属性更新，修改器

类似于 关系型，是否可以仅仅更新 文档的某个属性呢？使用修改器可以做到更新属性。

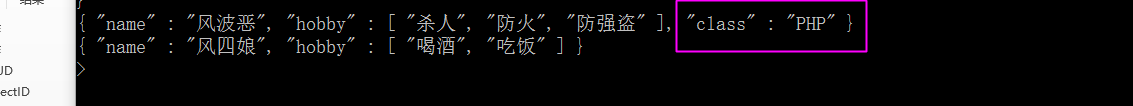
特殊的标识运算符：

#### $set，设置某个字段的值

{$set: {title: 'new-title'}}

存在则修改，不存在则添加。

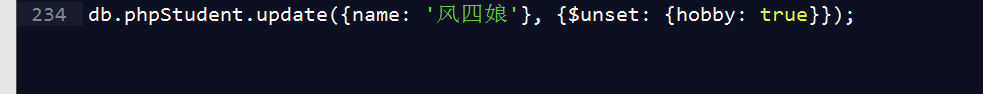




#### $unset，删除某个字段

{$unset: {class:true}}



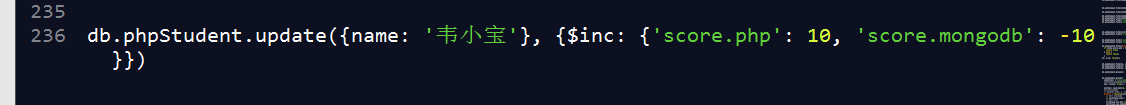


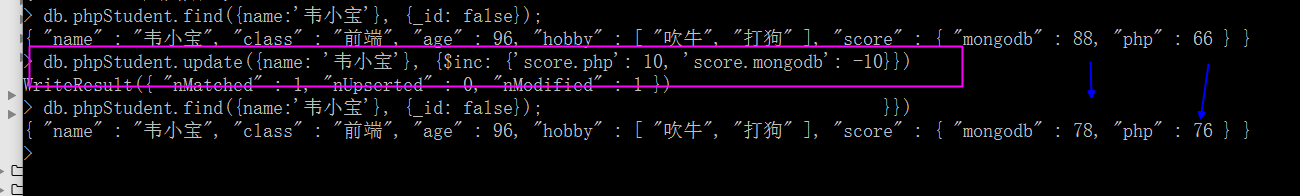


#### $inc，递增/递减

{$inc:{php:10}}递增

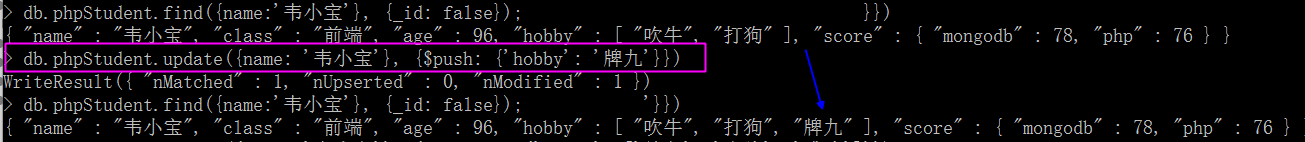
{$inc:{php:-10}}递减





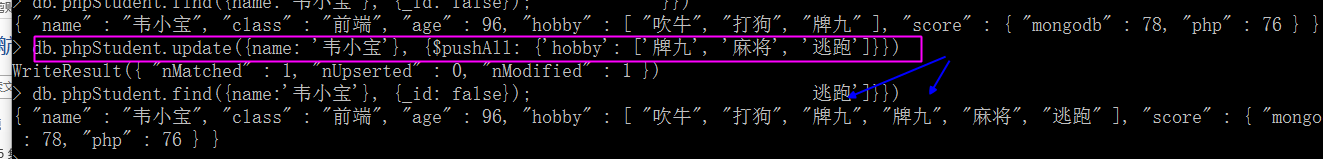
#### $push，数组入栈

{$push:{hobby:'new-hobby'}}



#### $pushAll，数组入栈多个元素

{$push:{hobby:['new-hobby-1', 'new-hobby-2']}}

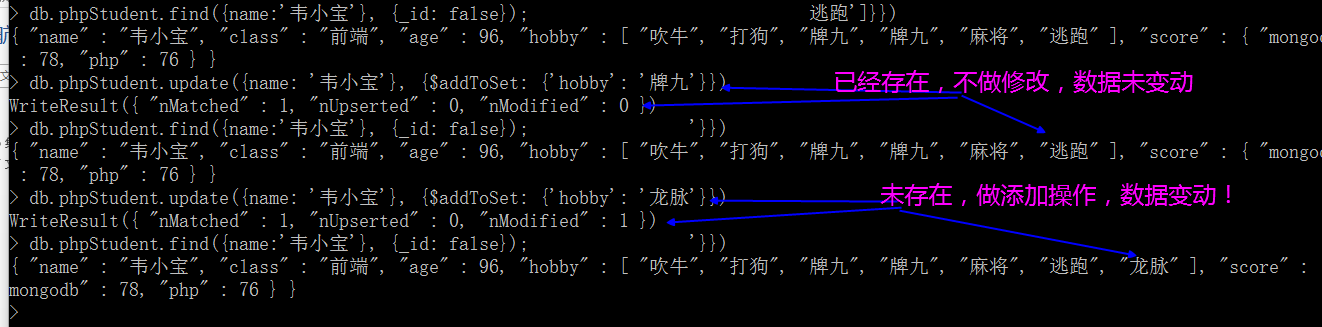


注意，无论push 还是 pushAll，是不会检测 数据元素的唯一性的。

#### $addToSet，加入到数组

{$addToSet:{hobby:'new-hobby'}}

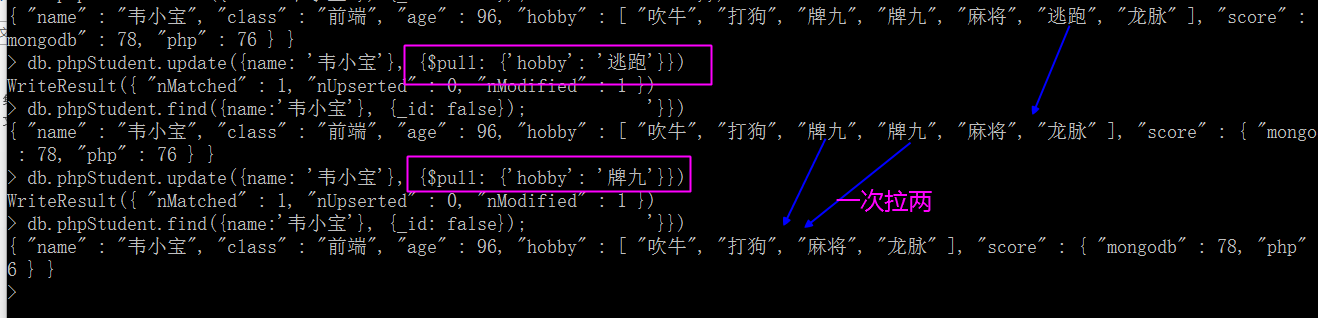
与push类似，加入集合前，会检测集合中的元素是否存在，如果存在不会重复添加。



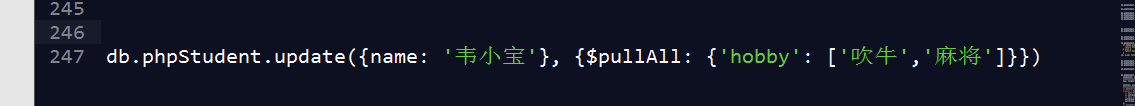
#### $pull，从数组中拉出数据

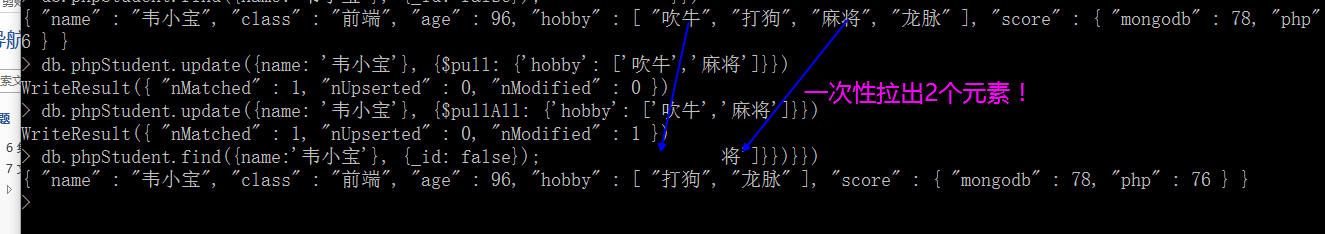
{$pull: {属性: ‘值’}}

匹配到数组元素，将其从数组中删除



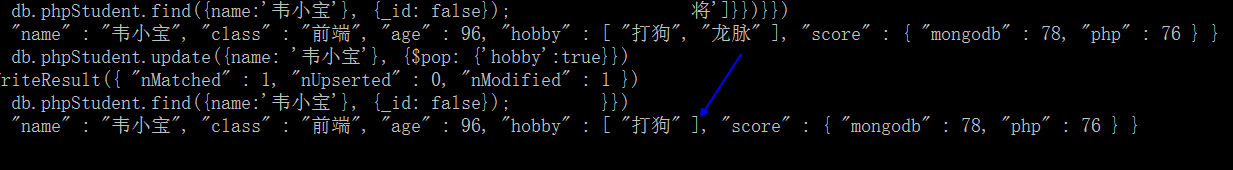
#### $pullAll，从数组中拉出多个数据



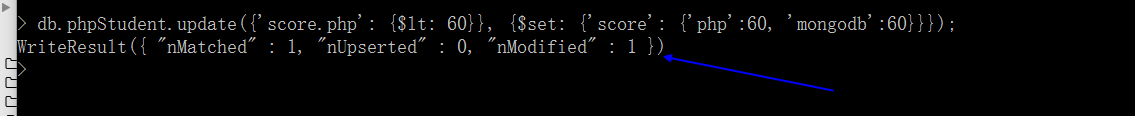


#### $pop，数组出栈

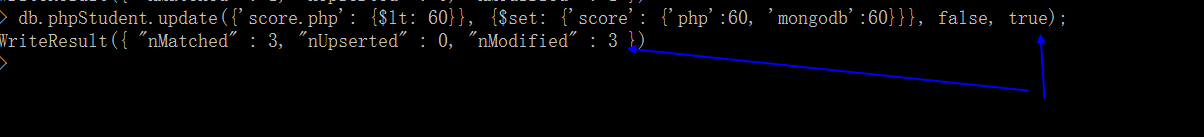
将数组中最后一个元素 出栈！



### 更新多条



需要使用第四个参数进行强行设置：



更新多条，目前要求配合修改器一起使用！



# 索引

加速排序，加速检索的 重要的功能！

在某个属性上加索引。

## 索引类型

### 普通索引，对关键字没有要求

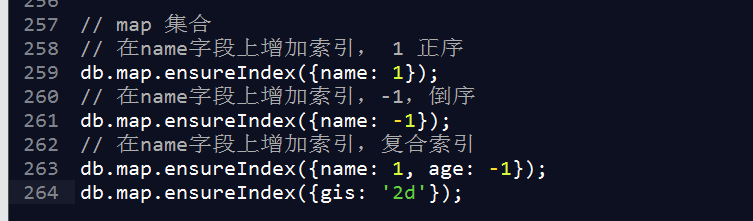
### 唯一索引，要求关键字唯一

### 2d平面空间索引，要求关键字是，坐标。x,y

## 管理索引

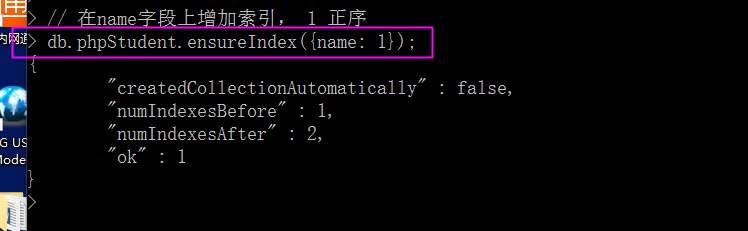
### 添加索引

db.collection-name.ensureIndex(属性集合，属性选项)



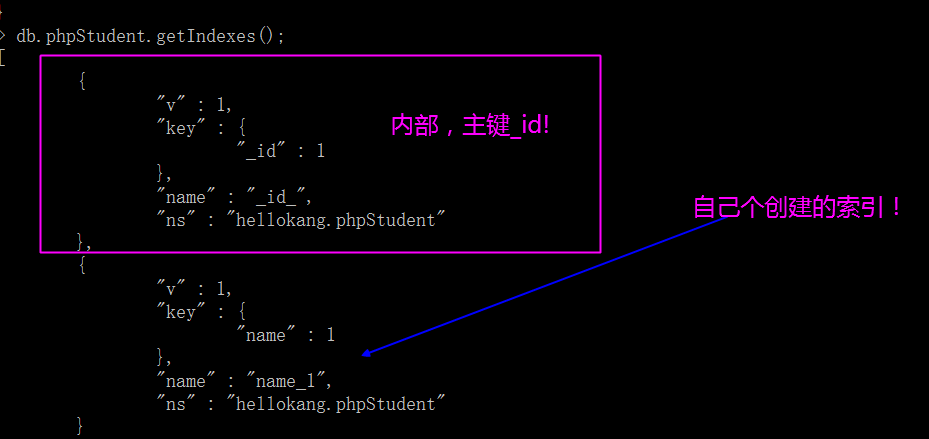
测试：

phpStudent上增加索引：



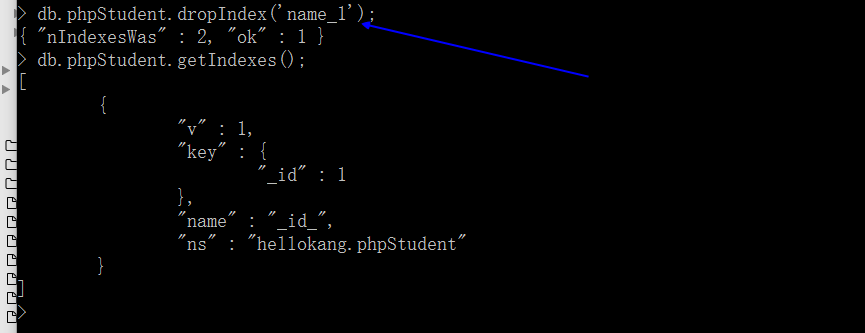
### 查看索引

db.collection-name.getIndexes();



### 删除索引

db.collectoin-name.dropIndex(‘index-name’);



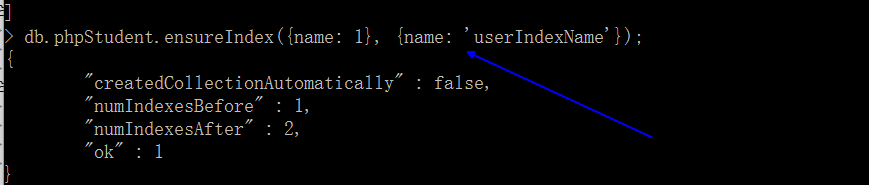
db.collection-name.dropIndexes()

删除集合上的全部索引。

只有在：初始化大量的集合数据时，才可能删除全部索引的。

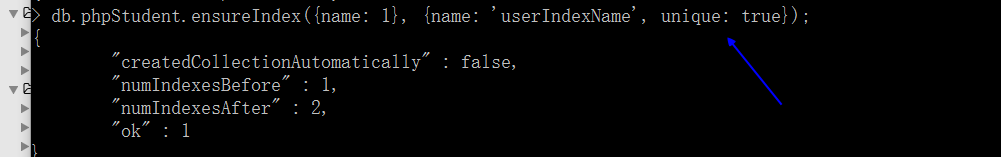
### 选项：指定索引名字

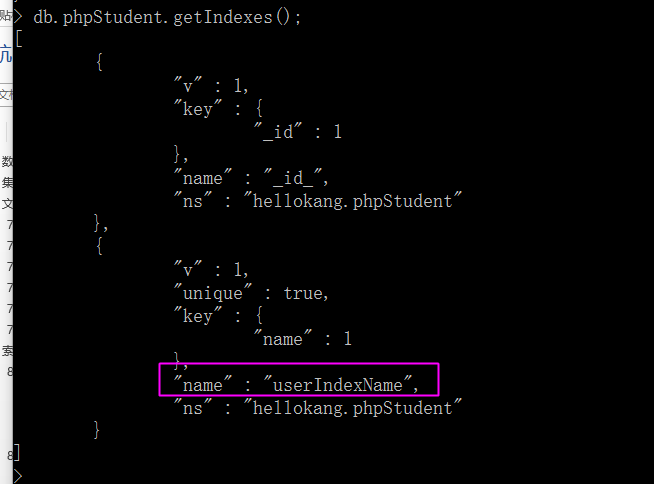
name选项：



### 选项：唯一索引

unique：true





要求：已经存在文档，文档的属性不能重复。

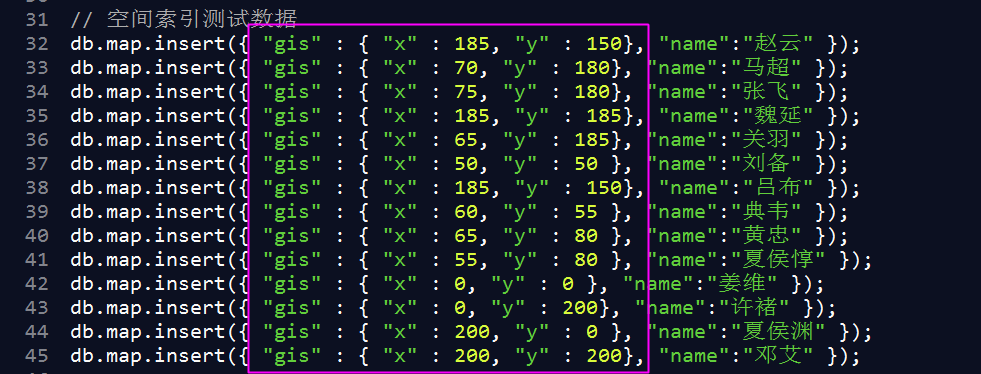
如果出现重复，选择选项：

dropDups:true，自动删除掉重复的文档！

## 平面空间2d索引

要求在 具有2个元素的属性上，创建2d索引：

例如：



对应：



将每个人的坐标，存储到mongo中。快速的检索：

1：谁离我近。距离检索。

2：范围内，谁在。范围检索。

实现：

增加文档数据：

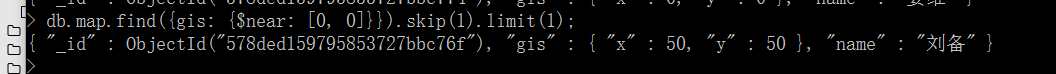
创建gis属性上的，空间二维索引



利用索引检索：

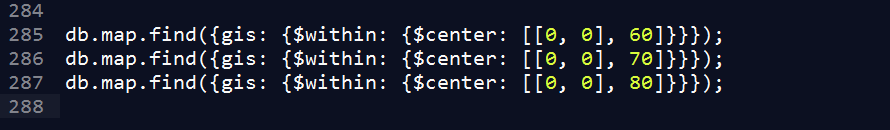
距离检索：{gis: {$near:[x,y]}}

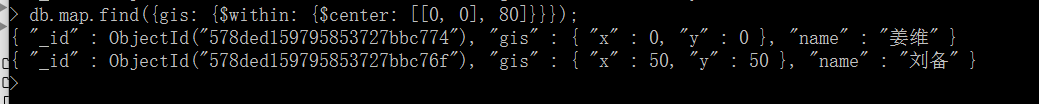
距离降维（0， 0）最近的人儿



范围检索：$within不同的形状，使用不同语法

圆形：$center





矩形范围：$box



# 功能概要

## 复制，replicate

master slave

## 分区（分片）

## map/reduce

大数据的特征。

大数据：

存储。分布式文件系统。（分片）

分析。map/reduce。

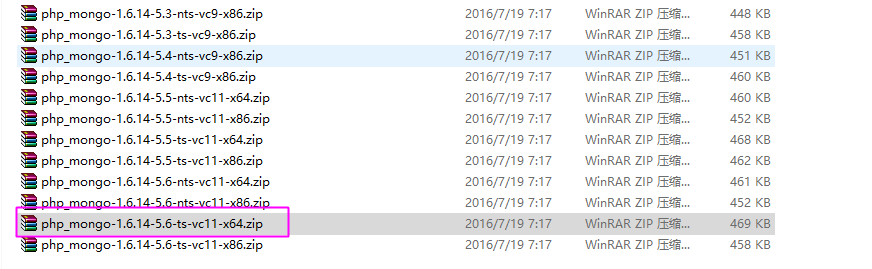
# PHP操作mongodb

PHP作为mongodb的客户端

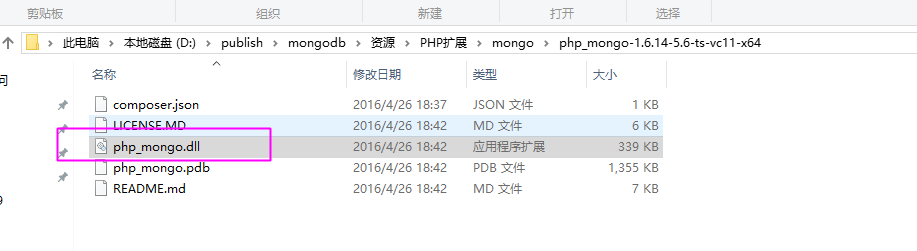
## 加载操作mongodb的扩展

mongo，mongodb

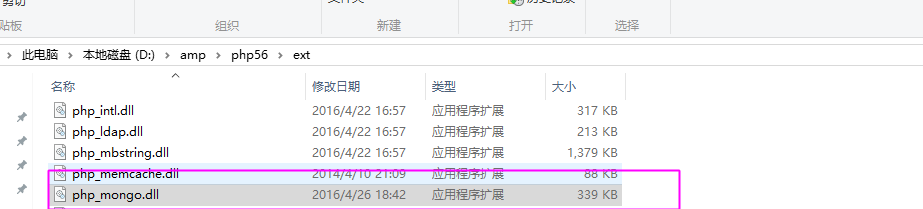
下载mongo.dll



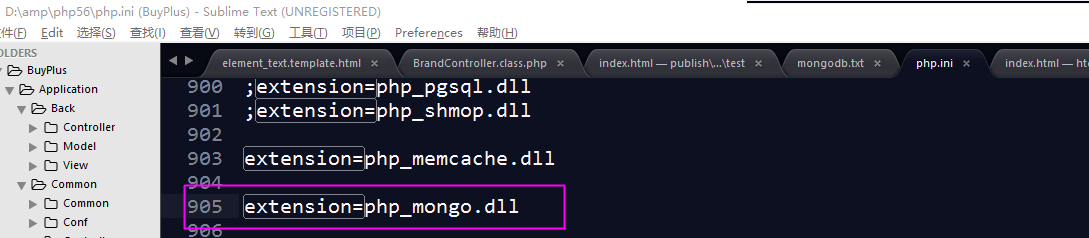
解压：



拷贝到PHP的扩展目录中

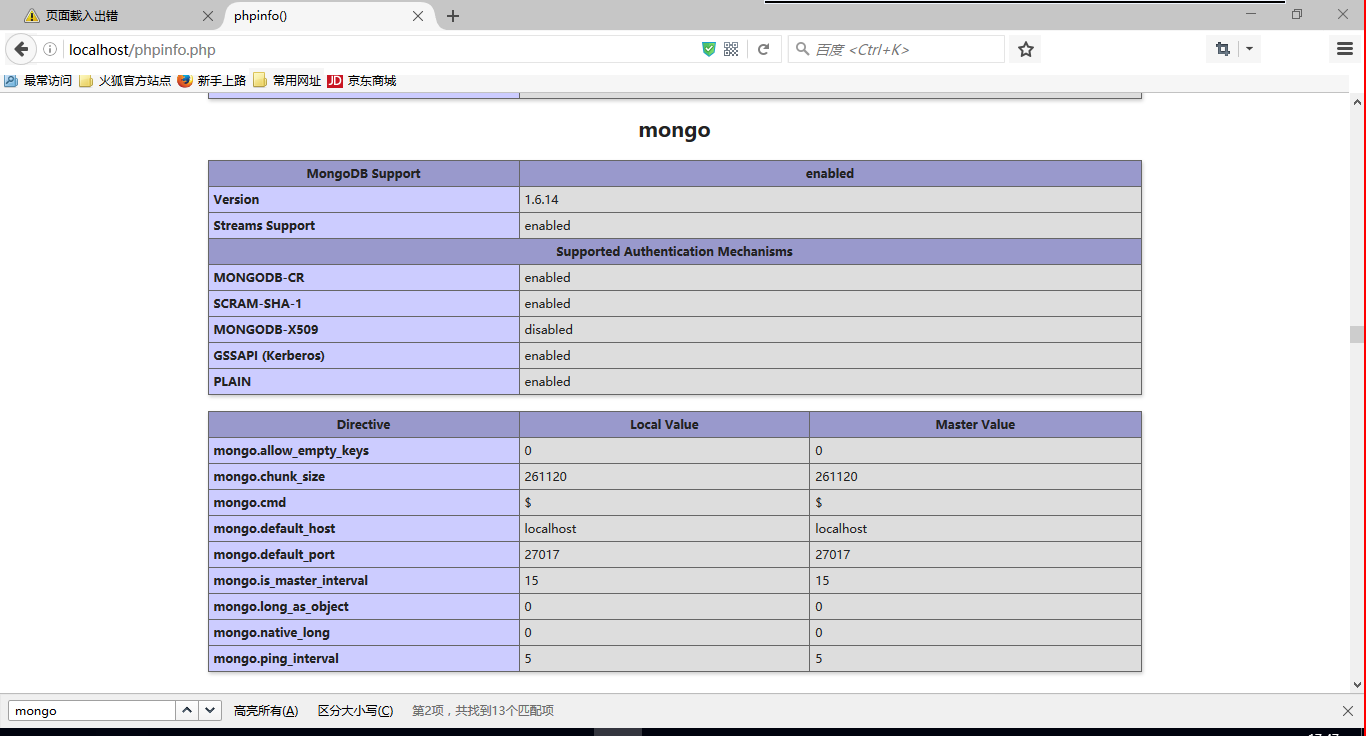


修改 php.ini



apache restart

测试



## 基本语法

面向对象语法。

提供的类：

MongoClient，连接服务器

MongoDB，操作数据库

MongoCollection，操作集合（常用）

MongoCursor，游标，处理数据

操作是无状态的。

典型的过程：

连接，

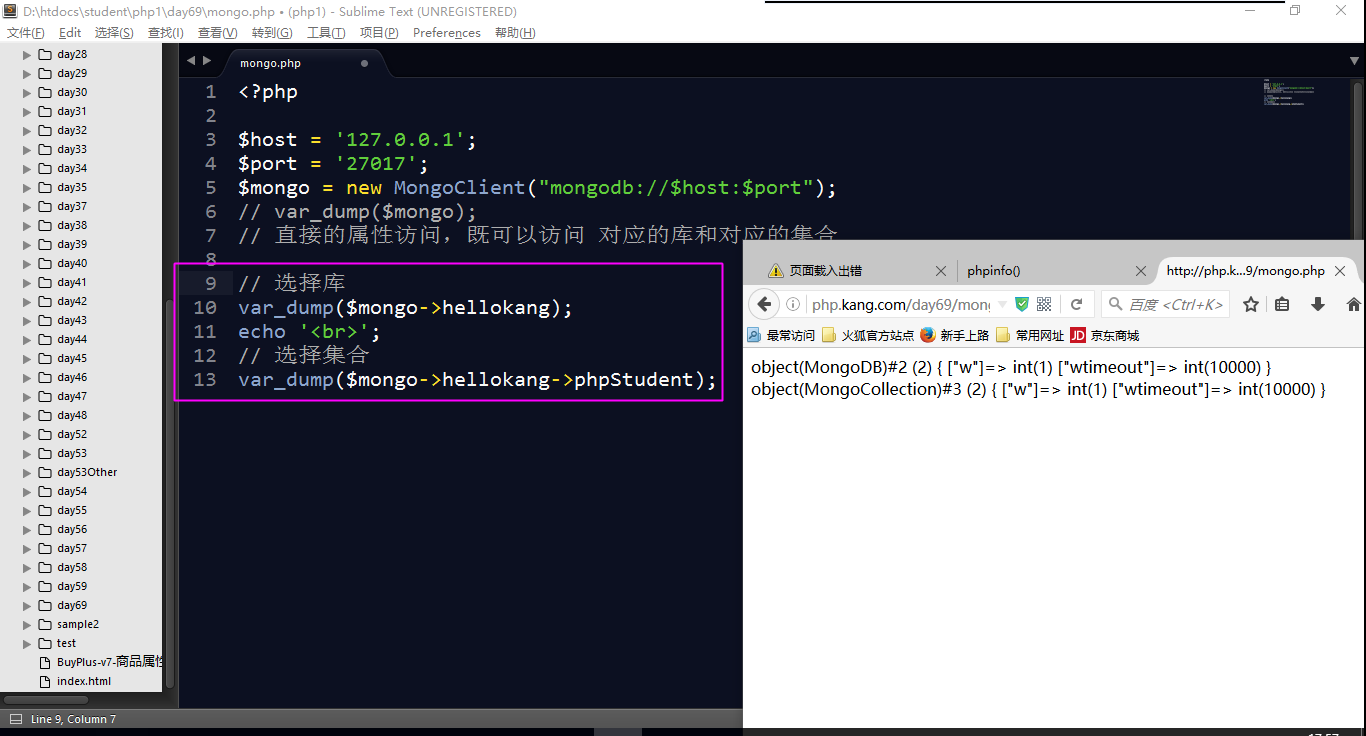
通过连接得到库，通过库操作集合，通过集合操作文档！

如果是获取查询文档，需要通过游标完成操作。

连接



操作，库，集合，文档



语法与mongo客户端语法，完全一致。

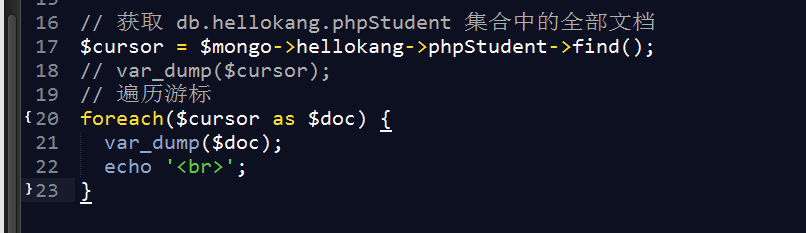
对应的转换：

JS数组：PHP索引数组

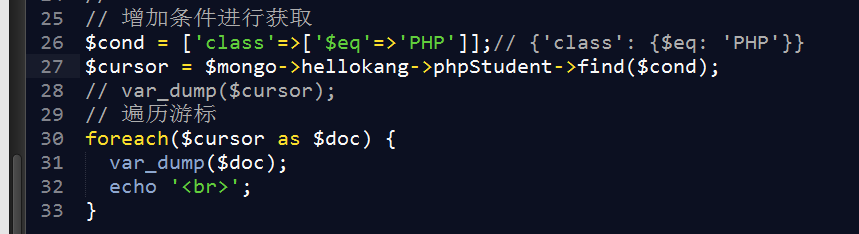
JS对象JSON：PHP关联数组

JS回调函数：PHP字符串（符合JS语法格式）

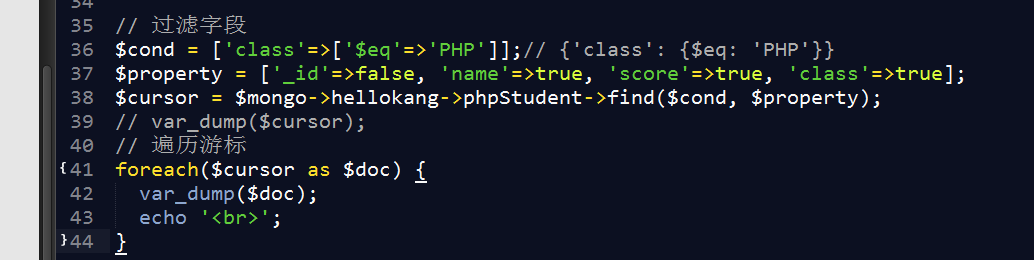
## 获取集合中的全部数据



## 增加条件获取

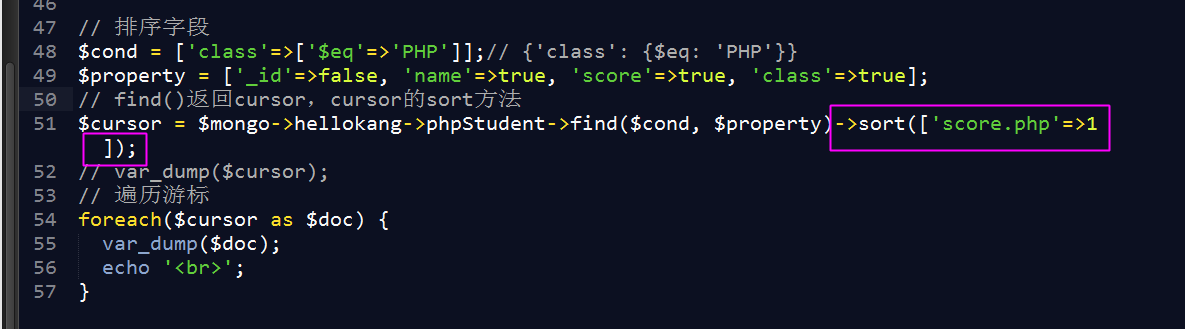


## 过滤字段



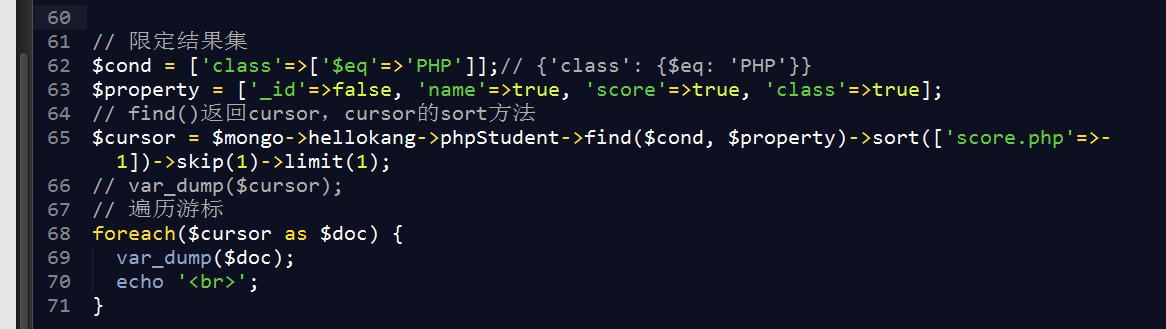
## 排序查询sort

sort(),游标的方法

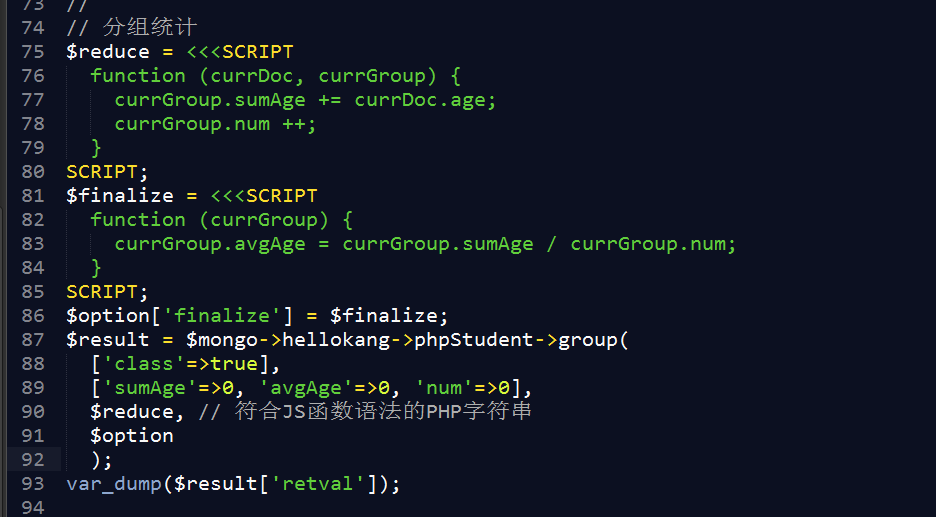


## skip()，limit()

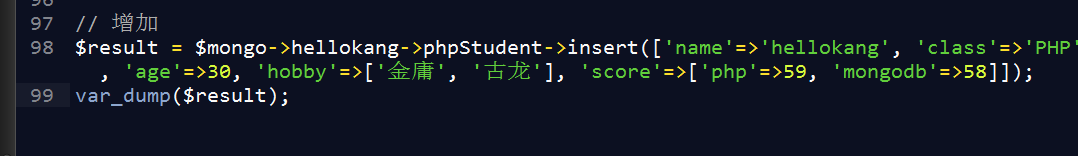
游标的方法



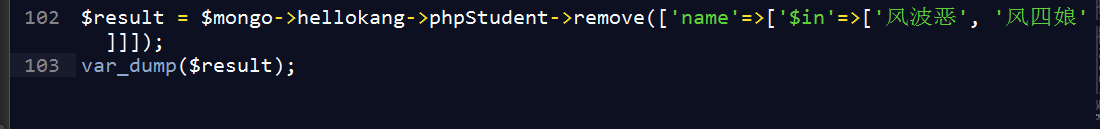
## 分组查询,group()



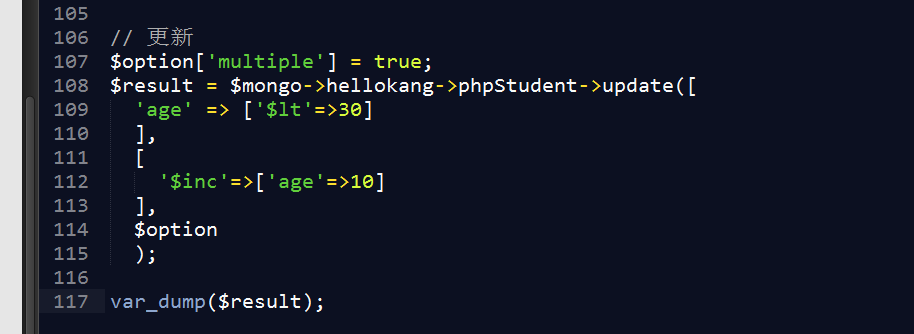
## 增



## 删除



## 修改



# mongodb的使用

## 哪里用?为什么用？

文档管理类。

## 不能用在哪里？

不处理关联，不处理频繁的更新。

## 数据结构的设计

article

category

tag

user

如果用mongodb管理：

article的集合，分类和tag，都是article的内联文档。

user 的集合