# 介绍

<https://www.mongodb.com/>

NoSQL:

Redis, memcached

* 键值（key-value）数据库 （redis，memcached）[存储用户信息，如会话]
* 面向文档（Document-Oriented）数据库 （mongodb，couchdb）[日志]
* 列存储（Column-Family）数据库（cassandra，hbase）[博客平台]
* 图（Graph-Oriented）数据库（orientdb，neo4j）[推荐引擎]

特点:

**MongoDB**是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。是非关系数据库当中功能最丰富，最像关系数据库的。支持的查询语言非常强大。

特点：

* 易使用
* 易扩展
* 易部署
* 高性能

# 安装

Mongod提供的是编译好的二进制文件包(不是源码包)

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

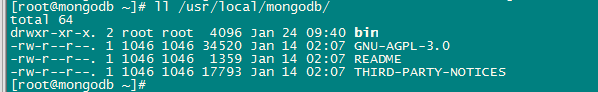
## Linux

CentOS 6.5 minimize

下载:

Wget https://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-2.6.7.tgz

解压到指定目录:



# 基本操作

## C/S架构

S: bin/mongod

C: bin/mongo

## 开启mongod守护进程

服务端 bin/mongod [--选项]

常用:

--dbpath 数据目录(必须要指定)

--logpath 日志文件地址

--fork 后台执行(以守护进程的方式运行)

--port 监听的端口(默认27017)

--bind\_ip 监听固定的网络IP接口(一台服务器多网卡,监听某个固定网卡的请求)

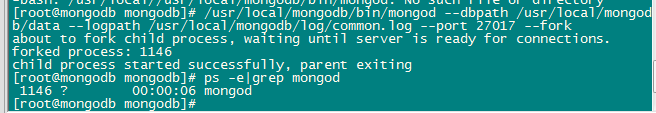
创建数据目录

Tip: 数据目录需要一定量的磁盘空间

mkdir /usr/local/mongodb/data

mkdir /usr/local/mongodb/log

运行:



提示: 可以简单的kill PID 关闭mongod守护进程

## 客户端bin/mongo



典型的服务器操作:

help

exit

# 操作

## 面向文档

关系型SQL: 数据库database, 表table, 记录row

面向文档: 数据库db, 集合collection, 文档document

MongoDB中的每个数据就是一个文档(document), 多个文档在一起组成collection

记录row PK 文档document:

记录row: 明确的字段，结构统一

文档document: key/value的集合(有序的), 但不要求同一集合中的文档key-value结构一致.

例如:

Student表:

黄蓉, 女, 打狗棒法

杨过, 男, 黯然销魂掌

小笼包, 女, 左右互搏术

Kang, 男, NULL

对应集合:

Student(collection)

{“name”:”黄蓉”, “gender”:”女”, “Kung Fu“: “打狗棒法”}

{“name”:”Kang”, “gender”:”男”}

{“age”: 30}

## JSON

Javascript Object Notation(JavaScript 对象表示法) 一种数据格式

Mongodb 的文档的格式语法. 每个mongodb文档就是JSON数据.

## BSON

二进制的JSON(可以执行的JSON)

## JavaScript Shell

bin/mongo客户端

Mongodb默认提供的客户端是一个javascriptShell(JS语法解释器), 通过在mongo上使用JS语法完成数据库操作.

JavaScript Shell通常提供:

1 JS核心语法的解释

2特定的对象

MongoShell提供的特定对象就是操作mongodb数据的对象:

**db 数据库对象**

**db.collection 某个数据库下的集合对象**

**cursor 数据光标对象(结果集对象)**

## 无模式

文档可以直接操作.

# 数据库DB操作

查看库:

show dbs;

选择库:

use dbname;



Tip: 无状态性导致库是不必要创建, 直接use即可

当前库(对象)

db



db对象上有很多操作

帮助:

db.help()

删除库



Tip: 文档管理的所有操作都是由db对象开始, 表示操作当前库的某个集合的某些文档.(其实是命名空间NS,namespace)

# 集合collection

show collections

db.createCollection(*集合名*)



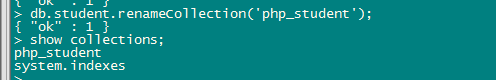
db.collection.help();



**use** mydb;

db.dropDatabase();

db.*collection*.renameCollection(*‘new name’*)

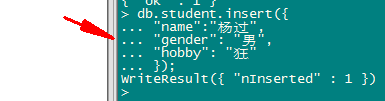


# 文档Document的CRUD

通过当前集合对象,方法完成集合内数据的增删改查.

插入

db.*collection*.insert(*json*);

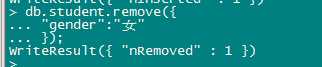


删除

db.*collection*.remove(*删除条件*)

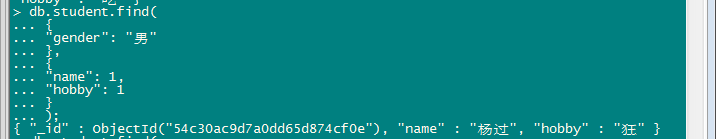
删除条件也采用json的语法.

例如:删除所有女生



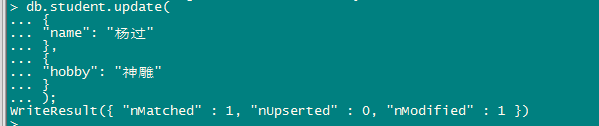
查看

db.*collection*.find(*查询条件, 返回属性列表*);



修改

db.*collection*.update(*更新条件, 更新数据*);

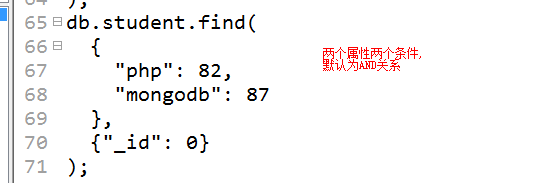


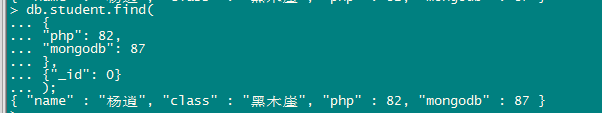


# 检索条件

Query

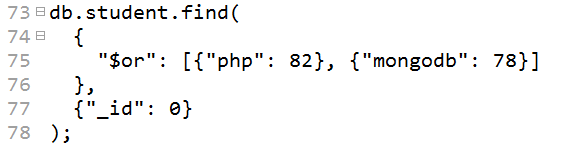
一个json对象, 每个属性一个条件





## 支持的逻辑关系

在mongo中,使用 $and, $or, $not 表示逻辑关系与,或,非



形成的条件: php=82 or mongodb=78



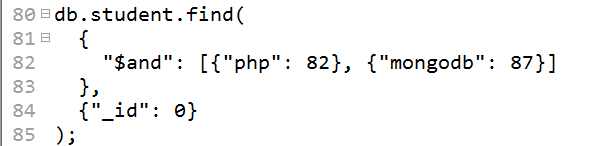
Tip: 记住上面的语法:

$ 开头就是mongo支持的运算符或指令

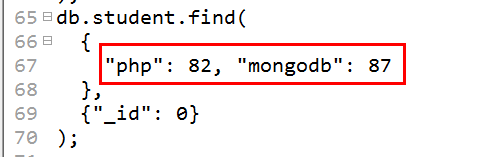
[] 数组的写法为特定的运算符传递的参数

一个元素, 一个条件

$and(默认)



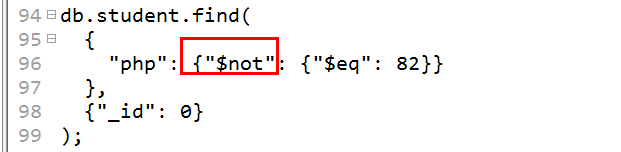
由于是默认关系,可以简写成:



$not

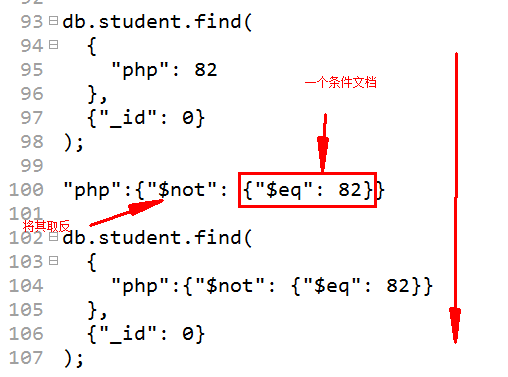
将结果取反,需要条件文档

例如: 取得 PHP成绩不为为82的学生:



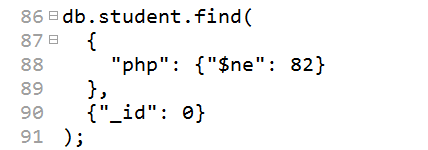
$eq: 一个关系运算符 =

过程如下:



Tip: 以上例子仅仅为演示 $not的用法:

取得 PHP成绩不为为82的学生逻辑来讲, 更简单的做法:



## 关系运算

通过关系元算符完成:

= $eq

!= **$ne**

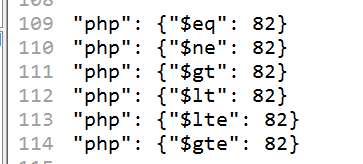
< $lt

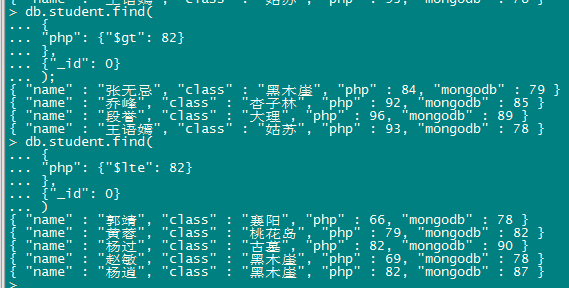
> $gt

<= $lte

>= $gte

属性对应一个表达式文档, 表达式文表示该属性与值的关系:



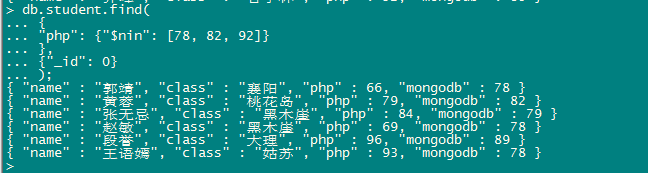


条件的写法支持嵌套:

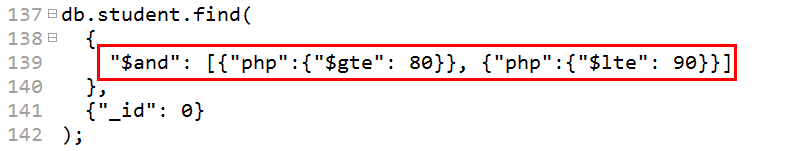


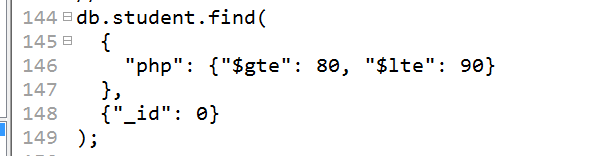
## $in/$nin





## 区间





## null, $exists



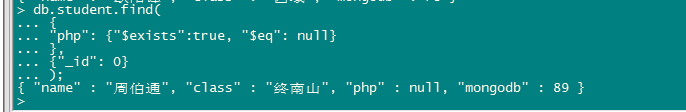
**{“property”: null}**

**可以匹配上属性为NULL的document. 同时,在文档中不存在的也会认为是null**

**通过 $exists提供布尔值匹配某个属性是否存在:**



**配合null使用就可以选出为NULL, 但是存在的数据:**

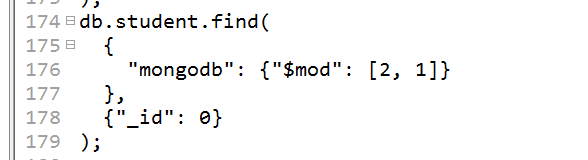


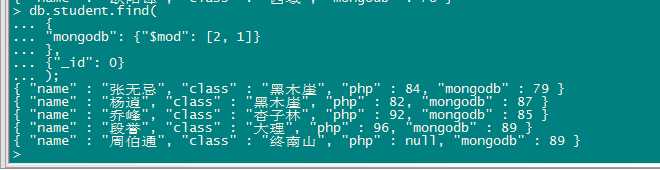
## $mod

求模

判断一个数据对D求摸后, 余数是否为M

{“$mod”: [D, M]}

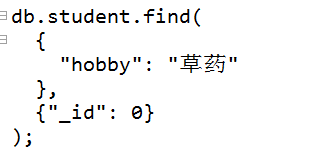




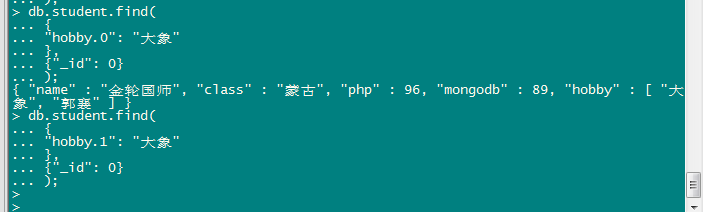
## 数组运算

如果属性的值为数组时, 使用数组运算操作, 完成条件的处理

直接匹配数组中的某个值:

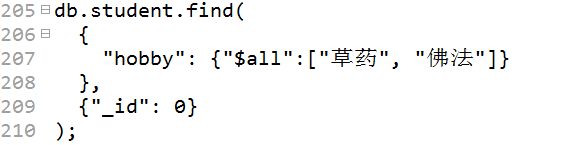


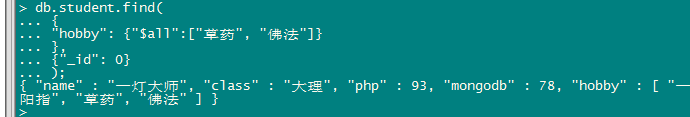
利用数组索引下标进行匹配:



$all

同时匹配 $all指定的所有数据(可以包含)





$size

匹配数组的size:

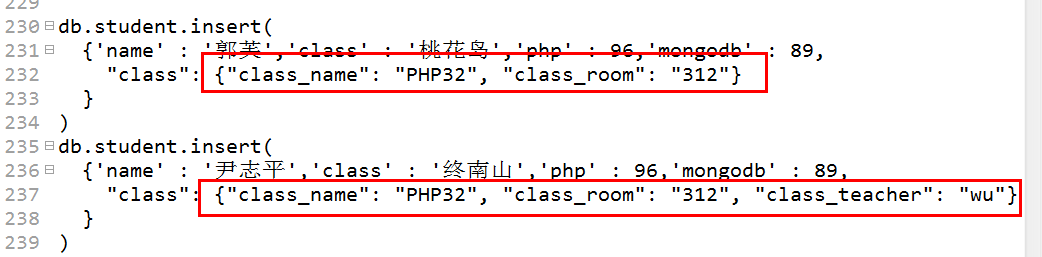
## 匹配正则



## 匹配内嵌文档(多层JSON)

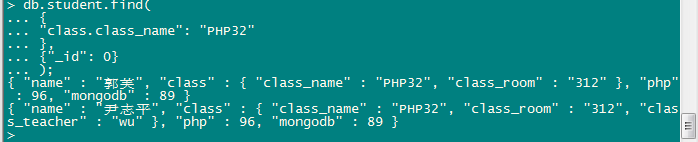
内嵌文档: 一个json文档中, 所包含的其他的json文档

{}属性键值列表就是一个json文档, 如果值又是json文档, 就称之为内嵌文档



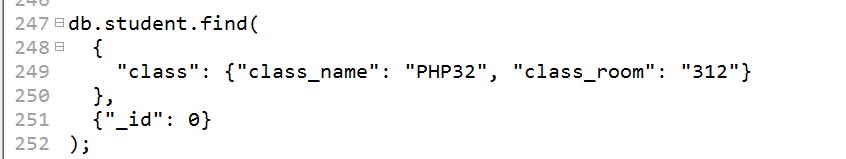
直接使用

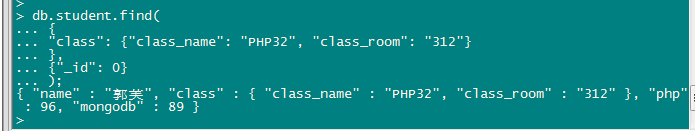
property.property的形式进行处理即可



Tip:

下面的语法将匹配值当作一个数据进行处理:





Tip:

内嵌文档, 就是mongodb用多级json 表示数据间关联的方式.

由于不支持连接join, 因此外键逻辑关联不可用, 内嵌文档就是mongodb表示实体联系的常用方式.

## $where

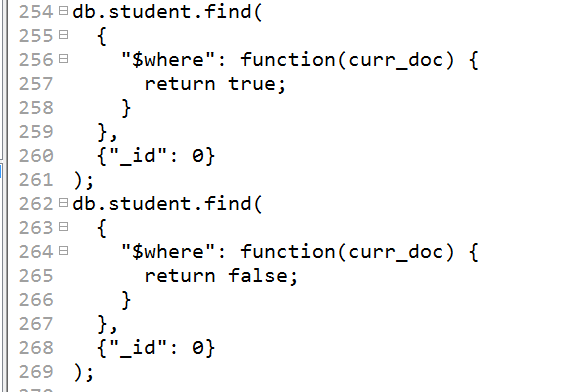
Mongodb不推荐使用 $where条件.速度较慢.

在使用key->value方式不能表示的情况下,才去使用 $where

{“$where” : function(curr\_doc){

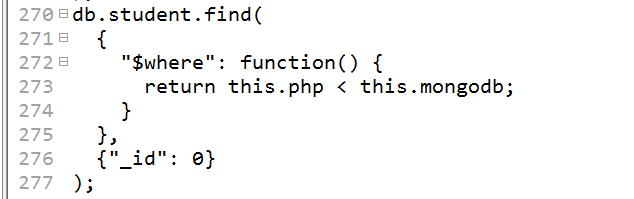
return true|false;

}}

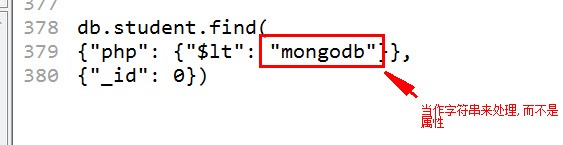


例如:

查找 php成绩小于mongodb成绩:



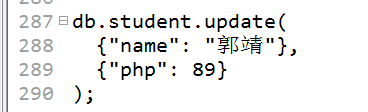
下面的做法就是错的:



# 更新的处理

db.*collection*.update(*条件, 更新文档*);

更新文档为一体的.

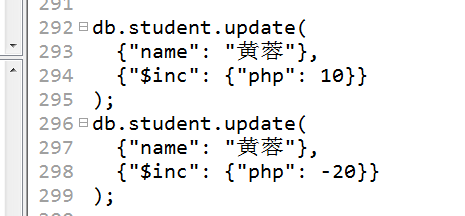


## 修改器(modifier)

如果需要仅仅修改某个字段, 则需要修改器(modifier):

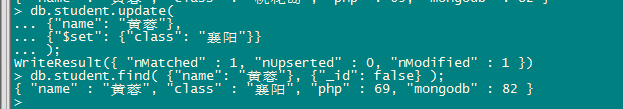
### $inc

递增/递减

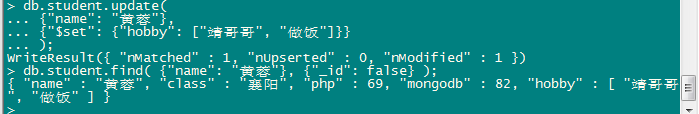


### $set

设置某个字段的值, 如果字段不存在则增加

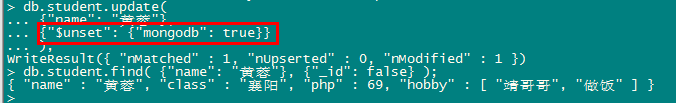


可以增加



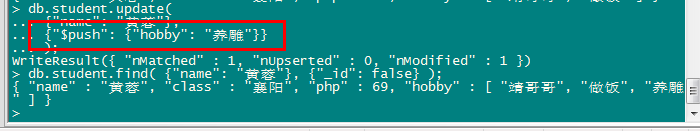
### $unset

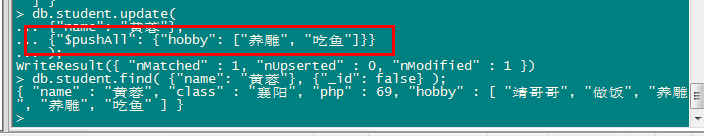
删除某个属性



### $push, $pushAll, $addToSet

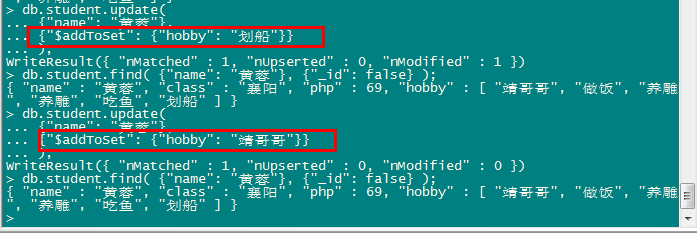
在数组值中, 入栈某(些)数组





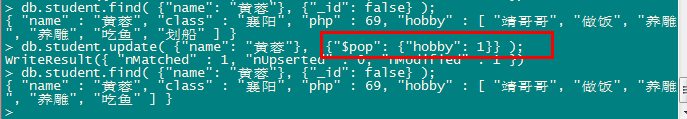
push/pushAll如果set中数据已经存在,也会成功压入.

使用 $addToSet 可以避免重复压入:



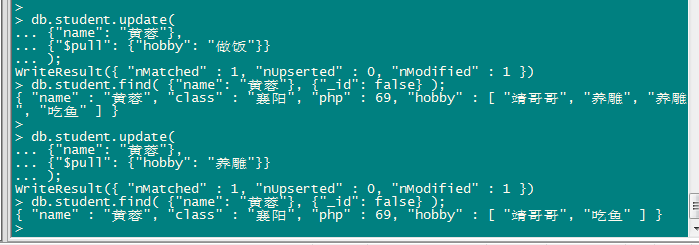
### $pop

出栈, 删除数组(集合)的最后一个元素



### $pull, $pullAll

从集合中去掉元素.

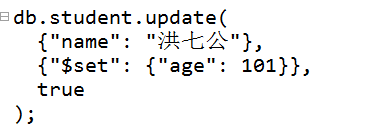


## upsert

update + insert

更新时插入, 如果没有匹配到数据, 则会插入.

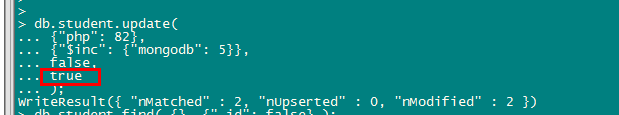
默认为false, 可以修改为true:





## multi

是否匹配多条记录, 默认的情况仅仅修改匹配第一条记录:



# 统计,分组,排序

## find().count()

find().count() 统计记录数



## group()分组检索

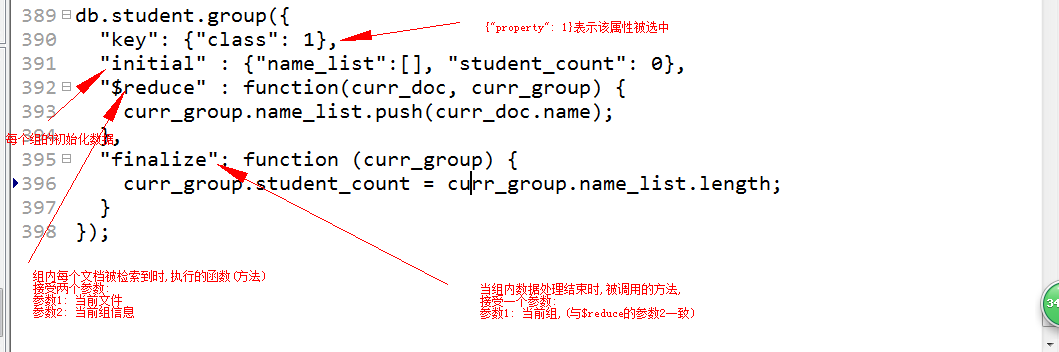
key ： 分组的key，

initial ：每组都分享一个 初始化 函数，

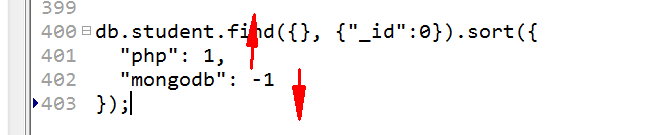
reduce：第一个参数表示当前文档对象，第二个参数是上一次function操作的累计对象。有多少个文档，$reduce就调用多少次。

condition：过滤条件

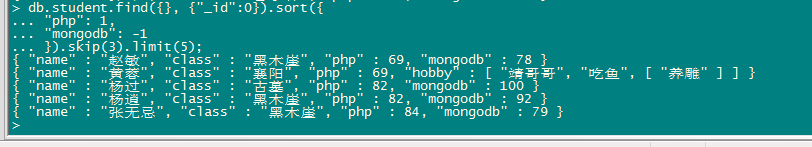
finalize：每一组文档执行完后，都会触发此方法，有一个参数，表示分组后的累计操作对象，和reduce中的第二个参数一致



## find().sort();



## find().skip().limit()



Tip:少用skip,用条件过滤去掉不需要的数据.

# 索引

为了高效检索数据:

三种索引:

普通索引,唯一索引,空间索引(坐标).

## 语法

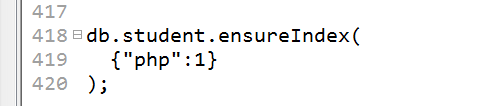
db.*collection*.ensureIndex(*{字段}, {选项}*);

db.*collection*.dropIndex(*索引名字*);

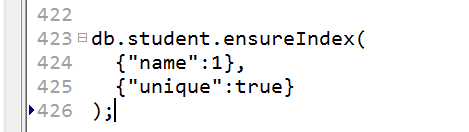
db.*collection*.dropIndexes();

db.*collection*.getIndexes()

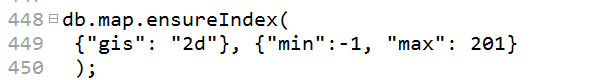
普通索引:



唯一索引:



空间索引:



演示空间索引的使用

操作符:

$near按照远近进行检索

$within按照范围进行检索

$box: 矩形

$center: 圆形

$polygon: 多边形

场景:

找到离夏侯惇最近的几个人:

先得到夏侯惇的坐标,在利用夏侯惇进行 $near检索:

{ "x" : 55, "y" : 80 }

查看索引:

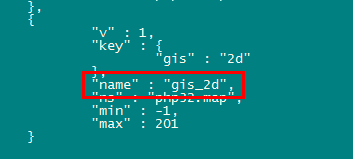
db.*collectoin*.getIndexes()

删除索引:

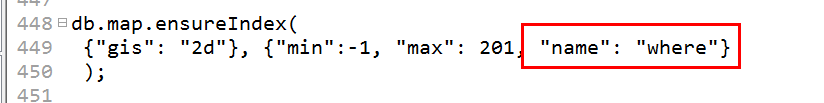
db.*collection*.dropIndex(*name*); //删除一个索引

db.*collection*.dropIndexes(); //删除当前集合的所有索引

索引的名字可以自动生成:

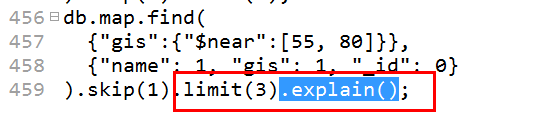


也可以在创建索引时指定:



find().explain()

语法可以解释当前查询的信息

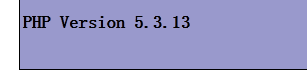


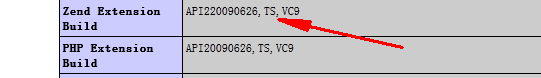
# PHP操作mongoDB

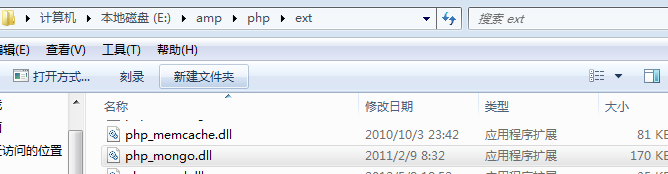
PHP作为mongoD的客户端

加载扩展, 使PHP支持mongo操作:

通过PHP版本选择合适的扩展dll



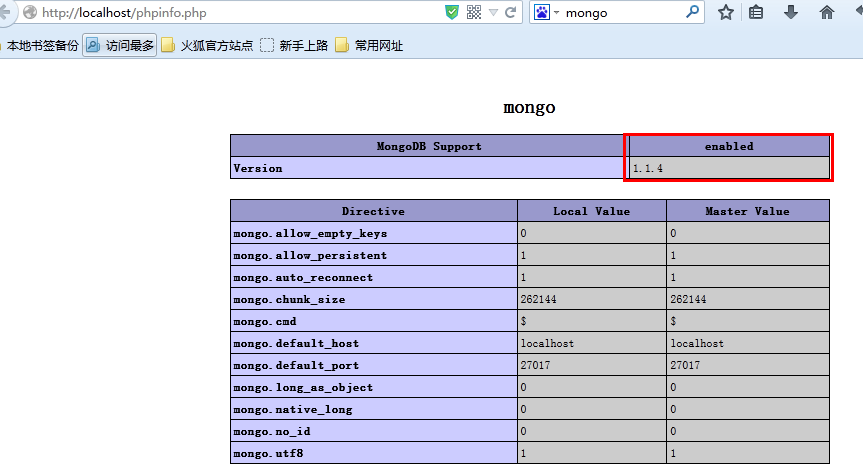




并开启扩展:

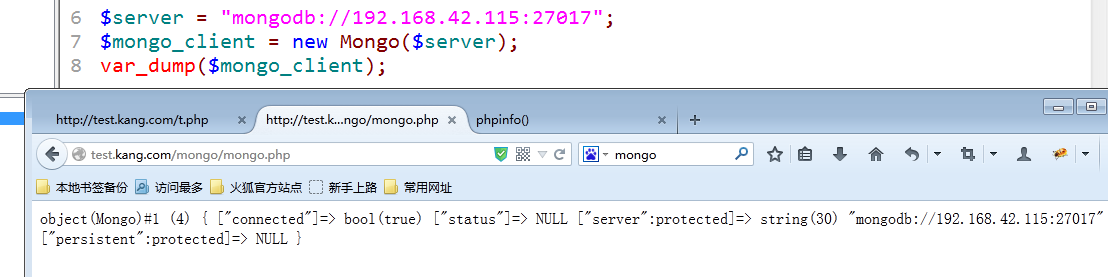
php.ini



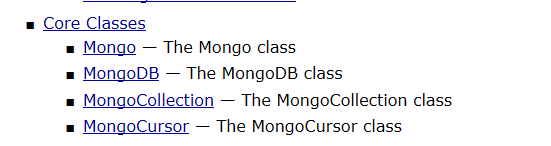


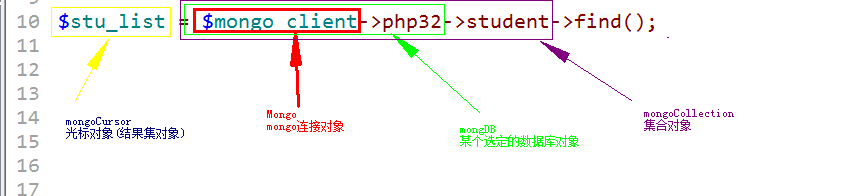
语法:

实例化 Mongo() 类对象进行操作即可:



当前mongo对象, 类似JS-shell中>,表示当前连接:





语法上,

所有的嵌套json 对应成嵌套的数组



# 功能列表

## 复制(主从)

利用复制做分布式集群,配合负载均衡软件完成

在开启服务时:

--master表示主服务器(写master)

--slave 表示从服务器(读) source masterIP:PORT

## 认证

Mongodb默认禁用认证机制

支持但没开启而已.

Db.createUser()创建用户

在开启服务器时

--auth选项表示该服务器支持认证

## 分区存储

支持跨服务器的分区存储.(云平台)

# Mongodb pk mysql?

关系型: 海量数据存储、维护数据关系完整性. 业务逻辑复杂,修改多的工作(OA, 管理平台)

mongoDB: 不考虑数据关系、更大的数据冗余, 更加高效读、插入.(文档的归类, 论坛帖子, 博客文章)