# 启动数据库

## 安装MongoDB

下载地址:

<https://www.mongodb.org/downloads?_ga=1.186672936.1287321877.1440466039>

## 启动MongoDB

分为三种方式

### Cmd命令行工具

服务器端工具 mongod

客户端工具 mongo

服务端启动指令:<local path>Mongod –dbpath <path>  
如本机的安装目录为D:\ToolsForDevelopment\MongoDB

则服务端启动为

客户端启动指令: <local path>mongo

同服务端启动

### 配置文件方式

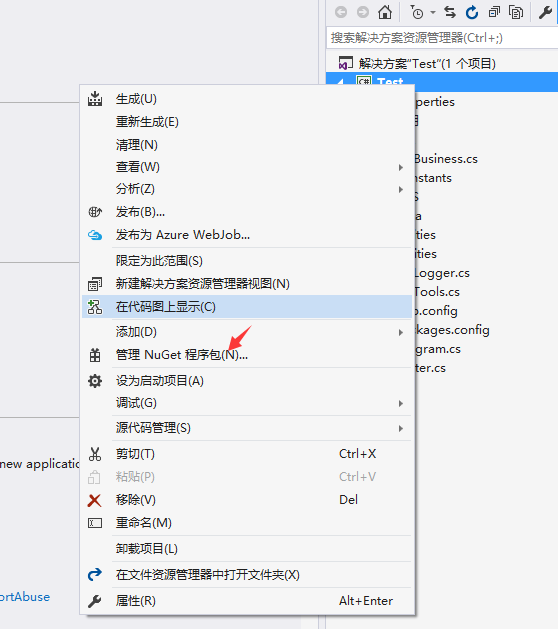
略

### Deamon方式

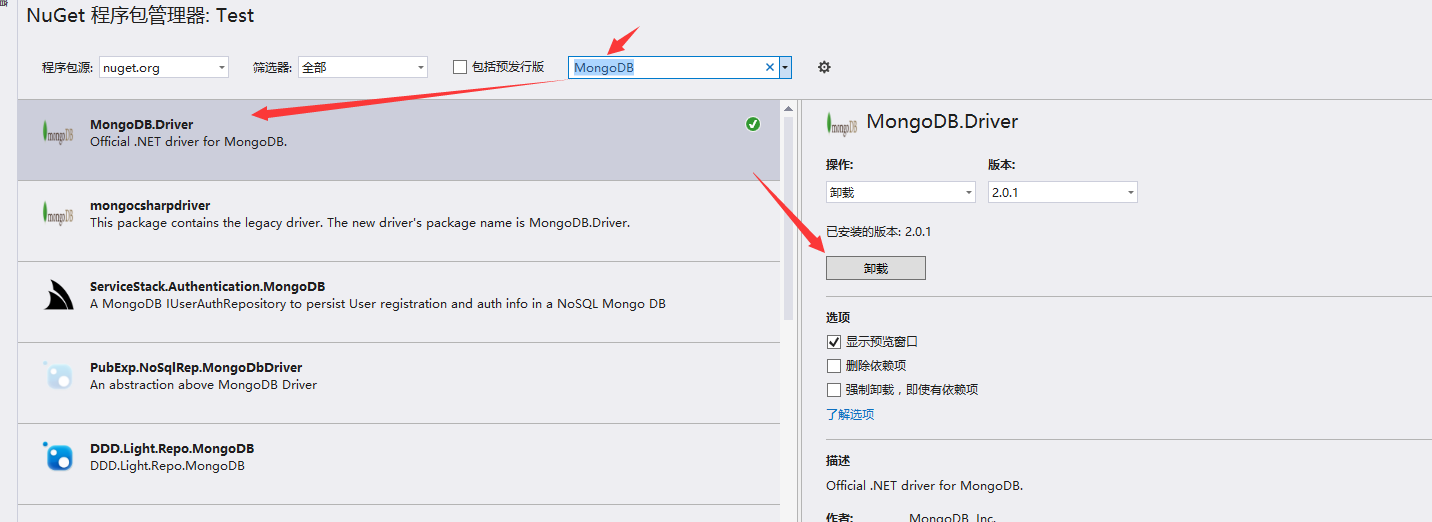
略

## .NET访问MongoDB

### Nuget下载驱动MongoDB

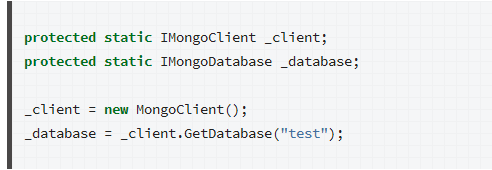


搜索MongoDB可简易安装MongoDB的官方驱动,下图本机为已安装状态.



### C#访问数据库示例:

<https://docs.mongodb.org/getting-started/csharp/client/>

# 指令 基础概念

**与关系型数据库对应的概念对象如下:  
元组 vs Document**

**表 vs Collection  
数据库 vs Database**

**MongoDB是一种模式自由的数据库**

**新增数据时可以直接插入,无需创建表的过程**

**数据库将自动创建对应Collection的存储文件**

**Document对象以{}表示**

**集合以[]表示**

## 获取帮助说明 db.help()

## 插入 db.<collectionName>.insert({<field>:<value>[,<field>:<value>]})

如新增张三  
db.student.insert({name:”张三”,age:22})

## 更新 db.<collectionName>.update({<fieldToMatch>:<value>},{<fieldToUpdate>:<value>})

如将张三年龄更新为23  
db.student.update({name:”张三”},{age:23})

## 删除 db.<collectionName>.remove({<fieldToMatch >:<value>})

如删除张三  
db.student.remove({name:”张三”})

## 查询 db.<collectionName>.find({<fieldToMatch >:<value>})

如查询所有年纪大于20的人,这里使用了$gt (greatThan操作符)  
db.student.find({age:{$gt:20}})

## 查询所有Collection db.getCollectionNames()

## 创建索引 db.<collecitonName>.createIndex()

具体索引的配置详情可查询官方资料.  
MongoDB提供了六类索引:单键索引(Single Field Index),复合索引(Compound Index),多键索引(Multikey Index),地理位置索引(Geospatial Index),文本内容索引(Text Index)以及散列索引(Hash Index)

以及三种索引特性:唯一(Unique),稀疏(sparse),生命周期(TTL)

## 查询所有索引 db.getIndexes()

# 操作符

略,详细内容请查阅官网:

<http://docs.mongodb.org/manual/reference/operator/>

# 表设计

## Schema

与关系数据库的模式不同

这里提供了两种方式

Reference vs Embed

### Reference(Normalized)

#### 适合多对多关系

#### 示例

db.student.insert({name:”张三”})  
MongoDB会为对象创建默认的索引字段\_id

var 张三 = db.student.findOne({name:”张三”})

db.student.insert({name:”李四”,friends:[new DBRef(“student”,张三.\_id)]})

这里创建了一个MongoDB中应用的引用对象DBRef  
创建规则new DBRef(<collectionName>,<\_id.value>)

引用对象可以通过fetch的形式获取

db.student.findOne({name:”李四”}).friends[0].fetch()

### Embed(Denormalized)

#### 适合一对一,一对多关系

#### 示例

db.student.insert(name:”张飞”,weapon:{name:”丈八蛇矛”,attack:12} )

其中weapon即是一个内嵌对象

## 设计要点

### MongoDB是一种文档数据库,故很多机制以文件的读写角度去考虑会更清晰

### 主旨:提供了灵活性和复杂度之间的权衡

### 充分利用Document Model的灵活性

#### 组织结构:Document,Collection,Database

#### 单Document尽可能的多数据,原子性存在于Document级别

### 考虑读写比例

#### 考虑Document对象大小

Document的大小限制(4MB or 16MB with 1.8),更大数据考虑GridFS

### 考虑读与更新的方式

### 考虑数据的生命周期和增幅