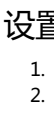


已久仰NoSQL的大名，知道它相对有关系型数据库，有很多的优点，只是一直没有时间来研究这个东西，所以借这个项目，对Mongodb进行了一次深入了解。



今天在李阳老师的指导下，八位同学开始nodejs研究的道路。



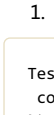
单独使用方法

```
npm install -g xxxxxx --registry=https://registry.npm.taobao.org
```

设置全局配置

1. 打开npmrc文件（在用户主目录下，没有的话可以建一个: touch ~/.npmrc）
2. 加入以下配置信息：

```
registry = http://registry.npm.taobao.org
```



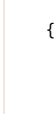
通过CMD生成文本目录，一般有下面两种方法：

1. 通过dir命令

```
dir /o /n /B
dir /o /n /B > file.txt //管道输出
```

2. 通过tree命令

```
tree /A
```



1. 查询

```
TestModel.find({'age' : 28},function(error,data) {
  console.log(data);
})
{} : 无查询参数时默认查出表中所有数据
```

2. Model保存数据

```
TestModel.create({
  name : "test_create",
  age : 26,
  email: "tom@qq.com"
},function(error,doc){
  console.log(doc);
});
```

例子

```
TestModel.create({
  name : "test_create",
  age : 26,
  email: "tom@qq.com"
},function(error,doc){
  console.log(doc);
});
```

当数据没有填充完整时输出一下数据，没有填写的字段不会输出

```
{
  age: 26,
  time: Sun Jan 31 2016 05:54:48 GMT+0000 (UTC),
  __v: 0,
  name: 'model_create',
  _id: 56ada1a87c5c6a4f00d1dac5
}
```

3. entity保存方法,model调用的是create方法，entity调用的是save方法

```
var Entity = new TestModel({});
Entity.save(function(error,data){})
```

4. 数据更新

```
var mongoose = require("mongoose");
var db =mongoose.connect("mongodb://127.0.0.1:27017/test");
var TestSchema = new mongoose.Schema({
  name : { type:String },
  age : { type:Number, default:0 },
  email: { type:String },
  time : { type:Date, default:Date.now }
});
var TestModel = db.model("test1", TestSchema );
var conditions = {age : 26};
var update = {$set :{name : '小小庄'}};
TestModel.update(conditions,update,function(error,data){
  if(error) {
    console.log(error);
  }else {
    console.log(data);
  }
})
返回结果 : { ok: 1, nModified: 1, n: 1 }
```

5. 删除数据

```
var conditions = { name: 'tom' };
TestModel.remove(conditions, function(error){
  if(error) {
    console.log(error);
  } else {
    console.log('Delete success!');
  }
});
```

总结

1. 查询: find查询返回符合条件一个、多个或者空数组文档结果。
2. 保存: model调用create方法，entity调用的save方法。
3. 更新: obj.update(查询条件,更新对象,callback)，根据条件更新相关数据。
4. 删除: obj.remove(查询条件,callback)，根据条件删除相关数据。

6. 简单查询方法 ---过滤

```
//返回一个只显示 name 和 email的属性集合
//id为默认输出,可以设置为 0 代表不输出

TestModel.find({}, {name:1, email:1, _id:0},function(err,docs){
  console.log(docs);
});
```

7. 单条数据 findOne(Conditions,callback);

```
查询符合条件的数据，结果只返回单条
TestModel.findOne({},function(error,data){
  console.log(data);
})
TestModel.findOne({age:28},function(error,docs){
  console.log(docs);
})
```

8. 单条数据 findById(id, callback);

根据id取数据findById，与findOne相同，但它只接收文档的_id作为参数，返回单个文档。

```
TestModel.findById('obj._id', function (err, doc){
  //doc 查询结果文档
});
```

9. 根据某些字段进行条件筛选查询，比如说 Number类型，怎么办呢，我们就可以使用\$gt(>)、\$lt(<)、\$lte(<=)、\$gte(>=)操作符进行排除性的查询

```
Model.find({"age":{"$gt":18}},function(error,docs){
  //查询所有nage大于18的数据
});

Model.find({"age":{"$lt":60}},function(error,docs){
  //查询所有nage小于60的数据
});

Model.find({"age":{"$gt":18,"$lt":60}},function(error,docs){
  //查询所有nage大于18小于60的数据
});
```