

- Home (/blogs)
- News (/blogs)
- Mooc (/moocs)
- Admin

User



**Zoemings** FOLLOW (<https://github.com/zoemings/>)

User Image

## Mongoose的基本用法

发布于 2016年7月16日 1 个月前

### 1. 将Mongoose集成到项目中

```
npm install --save mongoose
```

### 2. 连接数据库

```
var mongoose = require('mongoose');  
mongoose.connect('mongodb://127.0.0.1:27017/blog');
```

### 3. 定义一个Schema ( 也就是Mongodb中的Collections集合 ), 更多字段类型, 请参考SchemaTypes

```
var userSchema = {  
  username: {type: String, required: true, unique: true},  
  password: {type: String, required: true}  
}
```

#### 4. 将Schema进行 “Model化”

```
var User = mongoose.model('User', userSchema );
```

#### 5. 增加记录

```
User.create({username: '张三', password: 'md5-pass'}, function(err, user){  
  if(!err){  
    console.log(user.username + ' 保存成功!');  
  }else{  
    console.log('数据保存失败: ' + err);  
  }  
});
```

#### 6. 修改记录

```
User.findOneAndUpdate({_id: req.params.userId}, {  
  username: newUsername  
}, function (err, raw) {  
  if(!err) {  
    console.log( '修改成功!');  
  }else{  
    console.log('修改失败');  
  }  
});
```

#### 7. 删除记录

```
User.deleteById(userId, function(err, doc){  
  if(!err){  
    console.log('删除成功');  
  }  
});
```

#### 8. 查询记录

```
User.findById(userId, callback);    // one record  
User.findOne({username: '张三'}, callback);  // one record  
User.find(); // multi records
```

#### 9. 查询记录集合

```
User.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
```

#### 10. 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){})
```

#### 11. 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 }

```

## 12. 删除数据

```

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

```

## 13. 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);
});

```

## 14. 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);
})

```

## 15. 单条数据 findById(\_id, callback);

```

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

```

16. 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
});
```

## 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。

[read more](#)[导出pdf](#)