# MongoDB

Mongo - это noSQL БД. NoSQL BD делятся на несколько типов:

БД с типом записей в виде документов - такие, как MongoDB

БД с типом записей в виде ключ-значения, такие как Redis. Фишка - скорость работы

БД с типом записей в виде колонок, такие как Cassandra

БД с типом записей в виде графов, такие как Neo4j

В первом типе БД каждый кусочек информации - это **документ**. Записи могут в разных форматах, самый частоиспользуемый - JSON.

Документы могут быть объединены в **коллекции** документов.

Наобор коллекций документов и есть база данных.

## Монго сохраняет данные на диске в BSON (Binary JSON):

хранит префикс с длинной каждого значения - позволяет быстро пропускать ненужные поля, т.к. известна длинна конкретно нужного поля

хранит иформацию о типе поля - строка, число и др

хранит objectID

ObjectId - каждому документу присваивается свой ID. Если ID не указан специально, Монго даст ему свой. Обьект в Монго выглядит следующим образом:

{

"\_id": ObjectId("56ce74c0b0280ad12nc323xsad23),

"name": "Ultrapizza",

"description": "Test"

}

ObjectId занимает 12 байт и состоит из:

timestamp - 4 байта

Machine Id - 3 байта

Process Id - 2 байта

Increment - 3 байта

id.getTimestamp() возвратит timestamp в ISO Date формате

# Отличия NoSQL от SQL

В SQL - таблица (table), в NoSQL - коллекция (collection)

В SQL - запись (row), в NoSQL - документ (document)

В SQL - колонка (column), и у каждой записи в таблице будут одинаковые колонки

В NoSQL - поле (field), у каждой записи в документе поля могут отличаться полностью

# Установка

Скачиваем с оф. сайта Монго, ставим следуя базовым инструкциям. В папке проекта или рядом с ним создаем папку mongo в ней создаем папку data. Заходим в нее, копируем путь.

Для windows: запускаем cmd от имени императора, переходим в папку с Монго Сервером:

cd C:\Program Files\MongoDB\Server\3.2\bin

Дальше нам нужно указать для монго, что надо бы создать БД в папке с проектом, создаем там файл mongo, в нем файл data, пишем:

mongod.exe --dbpath="D:\Git\my\_notes\server\_side\node-examples\mongo\data"

После этого Монго сразу распакует в папку data свои файлы. Если че-то не получилось, смотреть тут <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

# Использование

НЕ ЗАКРЫВАЯ предыдущее окно, открываем новое, там пишем:

cd C:\Program Files\MongoDB\Server\3.2\bin

mongo.exe

Это откроет консоль управления базой данных

Команды

show dbs - показывает все БД

use conFusion - создает и переходит в новую БД conFusion. РЕГИСТРОЗАВИСИМ!!

db - показывает текущую БД

show collections или show tables - выведет имена коллекций

db.help() - выведет методы

Создать документ dishes и вставить в него данные:

db.dishes.insert({name: "Ultrapizza", description: "testing pizza"});

db.dishes.find(); - выводит инфу по документу dishes

db.dishes.find().pretty(); - выводит инфу в табличном виде

Создать новую переменную, посмотреть на нее, получить время

var id = new ObjectId();

id

id.getTimestamp();

Изменение (UPDATE) существующей записи

db.dishes.update({name: "Ultrapizza"}, {$set:{description: "changed description"}});

первое значение - уникальное имя записи, которую будем менять

второе - что меняем, в обьекте

## id

То, что отличает NoSQL от SQL БД.

Составной id позволяет с легкостью масштабировать приложения при увеличинии нагрузки, т.к. нет необходимости вычислять, какой id меньше или больше другого

id составляется из парметров:

* первые 4 байта - это timestamp, потому нет необходимости специально указывать время добавления записи
* следующие три байта - идентификатор машины, на которой запущена ДБ
* следующие 2 байта - идентификатор процесса
* последние 3 байта - и есть идентификатор записи, то, что делает и SQL

result.ops[0].\_id.getTimestamp() - покажет дату добавления в формате Timestamp

Создать новую переменную, посмотреть на нее, получить время

var id = new ObjectId();

console.log(id);

id.getTimestamp();

# Установка robomongo - GUI для mongo

Заходим на <https://robomongo.org/download>

Качаем, устанавливаем по дефолту, открываем

Открывается окно, жмем там строчку **Create**

Name **-** назовем как-нибудь, например Local Mongo Database

Остальные поля не трогаем, если монго установлен локально

Двойной клик откроет древовидное дерево папок

# Mongo DB native - модуль для Node.js

У Монго есть свой драйвер для Ноды. Создаем папку, назовем ее например node-mongodb - это и будет папка проекта. Заходим в нее, создаем папку node\_modules, копируем путь, переходим по нему в cmd и пишем:

npm install mongodb --save

**Пример 1.**

Создаем файл mongo-connect.js, в него пишем

const MongoClient = require('mongodb').MongoClient;

// 1st arg - url where DB lives, 2nd - callback

MongoClient.connect('mongodb://localhost:27017/TodoApp', (err, db) => {

if(err){

return console.log('Unable to connect to MongoDB server');

}

console.log('Connected to MongoDB server');

// insert new document into collection

db.collection('Todos').insertOne({

text: 'Something info',

complete: false

}, (err, result) => {

if (err) {

return console.log('Unable to insert Todo', err)

}

// show all docs

console.log(JSON.stringify(result.ops, null, 2));

// show timestamp

console.log(result.ops[0].\_id.getTimestamp());

});

db.close();

});

Это создаст базу данных TodoApp по указанному адресу. База создается автоматически. **Важно!** если в БД не добавлено никаких данных, БД не будет видна (не создастся).

Тут мы сразу добавляем коллекцию Todo, в которую добавляем документ с двумя свойствами

Сразу после добавления через insertOne в консоли отобразиться добавленный при помощи **result.ops**. JSON.stringify нужен для красивого и понятного отображения с отступами.

**Важно!** result.ops покажет добавленные только что документы, а не всю таблицу целиком!

## Пример 2, с модулем assert (проф. Muppala)

npm install assert --save

Модуль ассерт так же понадобится.

Создаем файл рядом simpleserver.js. В нем пишем:

var MongoClient = require('mongodb').MongoClient;

var assert = require('assert');

// URL для соединения, тут же указано имя БД

var url = 'mongodb://localhost:27017/conFusion';

// Метод для коннекта к серверу

MongoClient.connect(url, function (err, db){

// Выведет ошибку в консоль, или сообщение - connected correctly

assert.equal(err, null);

console.log("Connected correctly to server");

// создаем новый документ из вызванного обьекта db через callback

var collection = db.collection("dishes");

// инсертим первую пару ключ:значение в документ

collection.insertOne({name: "Ultrapizza", description: "test"},

function (err, result){ // callback функция

assert.equal(err, null); // ошибка в консоль

console.log("After Insert:"); // или сообщение

console.log(result.ops); // с массивом того, что было доабвлено

// после инсерта посмотрим, что лежит в документе

// вызывается специально в коллбеке, что бы док уже существовал

collection.find({}).toArray(function (err,docs){

assert.equal(err, null);

console.log("Found:");

console.log(docs);

// еще один колбек - удаление коллекции документов

db.dropCollection("dishes", function (err, result){

assert.equal(err,null);

db.close(); // закрыть соединение с базой данных

}); // end db.dropCollection

}); // end collection.find

}); // end collection.insertOne

}); // end MongoClient.connect

Запускаем Монго (БД conFusion уже должна быть создана, как выше), для этого открываем cmd и пишем:

cd C:\Program Files\MongoDB\Server\3.2\bin

mongod.exe --dbpath="D:\Git\my\_notes\server\_side\node-examples\mongo\data"

Открываем новое окно cmd (не закрывая окно, где запущено Монго!), запускаем сервер:

cd D:\Git\my\_notes\server\_side\node-examples\node-mongodb

node simpleserver

После чего должен появится результат в этом же окне