스타트업 개발자와 함께 공부하는

Node.js

09. 몽고 DB

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.

목차

NoSQL

NoSQL

등장 배경

- 빅데이터의 등장
- 비정형 데이터의 증가
- 유연한 스키마
- 분산 처리의 필요성

NoSQL

장단점

- ❑ 확장성
- ❑ 유연한 스키마
- ❑ 성능
- ❑ 고가용성
- ❑ 다양한 데이터 모델

- ❑ 일관성 문제
- ❑ 제한된 쿼리 기능
- ❑ 성숙도
- ❑ 관리성 복잡
- ❑ 표준화 부족

NoSQL

종류

- 문서 지향형 데이터베이스 – Mongo DB
- 키 – 값 스토어 – Redis
- 열 지향 데이터베이스 - Casandra
- 그래프 데이터베이스 - Neo4j
- 객체 데이터베이스 – db40
- 검색 엔진 – ElasticSearch

몽고 DB

몽고 DB

소개

- 문서 지향형(Document-Oriented) 데이터 모델
- JSON과 유사한 BSON(Binary JSON) 형식으로 데이터 저장
- 스키마리스(Schema – less) 구조
- 수평 확장성 - Sharding
- 높은 성능 – 인덱스
- 풍부한 쿼리 언어 – 필터링, 정렬 등
- 복제와 고가용성 – Replica Set

몽고 DB

장단점

- ❑ 유연한 데이터 모델
- ❑ 빠른 개발 속도
- ❑ 확장성
- ❑ 강력한 쿼리 성능
- ❑ 풍부한 생태계

- ❑ 일관성 문제
- ❑ 메모리 사용
- ❑ 복잡한 운영
- ❑ 학습 곡선

몽고 DB

주요 사용처

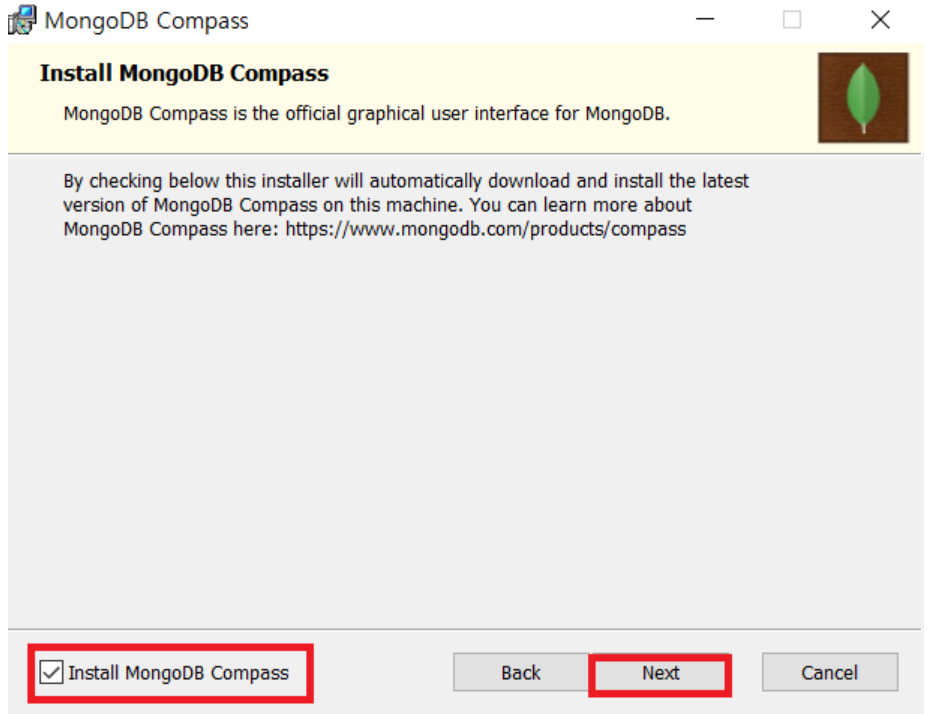
- 콘텐츠 관리 시스템 (CMS)
- 전자 상거래
- 빅데이터 애플리케이션
- IoT

몽고 DB 설치 및 연습

몽고 DB 연습

설치

- https://www.mongodb.com/try/download/community
- Windows x64 다운로드
- 설치파일 실행
- Install MongoDB Compass 체크
- 환경변수에 몽고 DB bin 디렉토리 추가



몽고 DB

구조

❑ 몽고 DB의 구조는 RDMBS 의 구조와 유사

몽고 DB	RDMBS
Collection	Table
Document	Row
Field	Column
Object ID	Primary Key
Link	Relationship

몽고 DB

기본 명령어

```
1  // 사용중인 DB 확인
2  db
3
4  // 데이터베이스 선택
5  use userdb
6
7  // 컬렉션 보기
8  show collections
9
10 // 컬렉션 생성
11 db.createCollection("users")
12
13 // 컬렉션 삭제
14 db.users.drop()
```

- [C]-[nodejs]-[project]-[09]-[ch09_01.mongodb]

몽도 DB

CRUD 명령어

// ch09_01.mongodb

```
14 // 데이터 삽입
15 db.users.insertOne({ name: "Alice", age: 30, city: "New York" })
16 db.users.insertMany([{ name: "Bob", age: 25, city: "Los Angeles" }, { name: "Charlie", age: 35, city: "Chicago" }])
17
18 // 데이터 조회
19 db.users.find({})  // 모든 문서 조회
20 db.users.find({ age: { $gt: 25 } })
21 db.users.findOne({ name: "Alice" })
22
23 // 데이터 수정
24 db.users.updateOne({ name: "Alice" }, { $set: { age: 31 } })
25 db.users.updateMany({ city: "Los Angeles" }, { $set: { city: "San Francisco" } })
26
27 // 데이터 삭제
28 db.users.deleteOne({ name: "Charlie" })
29 db.users.deleteMany({ age: { $lt: 30 } })
```

몽도 DB

추가 명령어

// ch09_01.mongodb

```
34 // 인덱스 생성
35 db.users.createIndex({ name: 1 })  // name 필드에 대해 오름차순 인덱스 생성
36
37 // 컬렉션 삭제
38 db.users.drop()
39
40 // 데이터베이스 삭제
41 db.dropDatabase()
42
43 // 정렬
44 db.users.find({}).sort({ age: -1 })  // 나이를 기준으로 내림차순 정렬
```

몽구스

몽구스

소개

- 몽고 DB 객체 데이터 모델링 라이브러리
- 스키마 정의
- 모델 생성
- 밸리데이션
- 쿼리 빌더

몽구스

프로젝트 생성 및 설치

```
npm init -y
npm install mongoose
```

❑ [C]-[nodejs]-[project]-[09]-[ch09_02]

몽구스

몽고 DB 연결 및 스키마 생성

```
// ch09_02.js
const mongoose = require('mongoose');

(async () => {
  await mongoose.connect('mongodb://localhost:27017/mydb');
  console.log('Connected to MongoDB');

  const { Schema } = mongoose;

  const userSchema = new Schema({
    name: { type: String, required: true },
    age: { type: Number, min: 0, max: 120 },
    city: String,
    email: { type: String, match: /.+\@.+\..+/ }
  }, { timestamps: true });

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

몽구스

기본 CRUD

```
// ch09_02.js
```

```
19    const newUser = new User({ name: 'Alice', age: 30, city: 'New York', email: 'alice@example.com' });
20    const savedUser = await newUser.save();
21    console.log('User saved successfully:', savedUser);
22
23    const users = await User.find({});
24    console.log('All users:', users);
25
26    const filteredUsers = await User.find({ age: { $gte: 25 } });
27    console.log('Users aged 25 and above:', filteredUsers);
28
29    const updatedUser = await User.updateOne({ name: 'Alice' }, { $set: { age: 31 } });
30    console.log('User updated successfully:', updatedUser);
31
32    const deletedUser = await User.deleteOne({ name: 'Alice' });
33    console.log('User deleted successfully:', deletedUser);
34
35  })();
```

답변형 게시판

몽구스

프로젝트 생성 및 설치

```
npm init -y
npm install express mongoose nodemon
```

- [C]-[nodejs]-[project]-[09]-[ch09_03]

- board.js
- db.js
- routes.js
- server.js

답변형 게시판

몽고디비 연결

```
// db.js
1  const mongoose = require('mongoose');
2  mongoose.connect('mongodb://localhost/facebook');
3
4  const db = mongoose.connection;
5
6  db.on('error', (err) => {
7      console.error(`db connect error : ${err}`);
8  });
9
10 db.once('open', () => {
11     console.log(`Database connected successfully `)
12 });
```

답변형 게시판

스키마 생성

```
// board.js
1  const mongoose = require('mongoose');
2
3  const BoardSchema = new mongoose.Schema({
4      title: String,
5      content: String,
6      writer: String,
7      write_date: { type: Date, default: Date.now } ,
8      comments: [
9          {
10             comment: String,
11             user: String,
12             created_at: { type: Date, default: Date.now }
13         }
14     ]
15 });
16
17 const Board = mongoose.model('Board', BoardSchema);
18
19 module.exports = Board;
```

답변형 게시판

글 작성

```
// routes.js
```

```
6  router.post('/boards', async (req, res) => {
7      const {title, content, writer} = req.body;
8
9      try{
10         const board = new Board({
11             title: title,
12             content: content,
13             writer: writer,
14         });
15         board.save();
16         res.status(200).json(board);
17     }catch(error) {
18         console.error(`post error : ${error}`);
19         res.status(200).json({error: error})
20     }
21 });
```

답변형 게시판

글 목록

```
// routes.js

23 router.get('/boards', async (req, res) => {
24     try{
25         const boards = await Board.find({});
26         res.json(boards);
27     }catch(error) {
28         console.error(`get error : ${error}`);
29         res.status(200).json({error: error})
30     }
31 });
```

답변형 게시판

글 상세

```
// routes.js

33 router.get('/boards/:id', async (req, res) => {
34     const { id } = req.params;
35     try{
36         const boards = await Board.findById(id)
37         res.json(boards);
38     }catch(error) {
39         console.error(`get error : ${error}`);
40         res.status(200).json({error: error})
41     }
42 });
```

답변형 게시판

글 수정

```
// routes.js
44 router.put('/boards/:id', async (req, res) => {
45     const { id } = req.params;
46     const { title, content } = req.body;
47 
48     try{
49         const board = await Board.findByIdAndUpdate(id, {
50             title, content
51         }, {new: true});
52         res.status(200).json(board);
53 
54     }catch(error) {
55         console.error(`put error : ${error}`);
56         res.status(200).json({error: error});
57     }
58 });
```

답변형 게시판

글 삭제

```
// routes.js
61 router.delete('/boards/:id', async (req, res) => {
62     const { id } = req.params;
63 
64     try{
65         const board = await Board.findByIdAndDelete(id);
66         res.status(200).json(board);
67     }catch(error) {
68         console.error(`put error : ${error}`);
69         res.status(200).json({error: error});
70     }
71 });
```

답변형 게시판

답변 등록

```
// routes.js
73 router.post('/boards/:id/comments', async (req, res) => {
74     const { id } = req.params;
75     const { comment, user } = req.body;
76     try{
77         const board = await Board.updateOne({
78             _id: id
79         }, {
80             $push: {comments: {comment: comment, user: user}}
81         });
82         console.log(`put comments : ${id}, ${cid}, ${comment}, ${user}, ${JSON.stringify(board)}`)
83         res.redirect('/boards');
84     }catch(error) {
85         console.error(`comments post error : ${error}`);
86         res.status(200).json({error: error});
87     }
88 });
```

답변형 게시판

답변 수정

```
// routes.js
```

```
90  router.put('/boards/:id/comments/:cid', async (req, res) => {
91      const { id, cid } = req.params;
92      const { comment, user } = req.body;
93      try{
94          const board = await Board.updateOne({
95              _id: id, 'comments._id':cid
96          }, {
97              $set : {
98                  'comments.$' : {
99                      'comment': comment,
100                     'user': user,
101                 }
102             }
103         });
104         console.log(`put comments : ${id}, ${cid}, ${comment}, ${user}, ${JSON.stringify(board)}`)
105         res.redirect('/boards');
106     }catch(error) {
107         console.error(`comments post error : ${error}`);
108         res.status(200).json({error: error});
109     }
110 });
```

답변형 게시판

답변 삭제

```
// routes.js
```

```
112 router.delete('/boards/:id/comments/:cid', async (req, res) => {
113     const { id, cid } = req.params;
114     try{
115         const board = await Board.updateOne({
116             _id:id,
117         }, {
118             $pull : {
119                 comments: {_id: cid}
120             }
121         });
122         console.log(`put comments : ${id}, ${cid}, ${JSON.stringify(board)}`)
123         res.redirect('/boards');
124     }catch(error) {
125         console.error(`comments post error : ${error}`);
126         res.status(200).json({error: error});
127     }
128 });
```

백문이 불여일타