Bagian Ke 8

MongoDB dan Mongoose Part 3

Pada materi ini kita akan belajar bagaimana mencari data di dalama collections, belajar bagaimana menemukan spesifik object didalam collections, kita juga akan belajar update dan remove data didalam collections.

Update halaman server.js

Halaman index:

```
// halaman index or root
app.get('/', function(req, res, next){
        res.json('Selamat Datang di Halaman
Beranda!')
});
```

Update menjadi

```
app.get('/', function(req, res, next){
        user.find({ name: "Dyo" }, function(err,
foundUser){
                if (foundUser){
                        res.json(foundUser);
                } else {
                        res.json("User tidak ada");
                }
        });
})
```

Test pada browser localhost:3000.

Hasilnya akan ditampilkan user dengan name = Dyo.

Coba sbb:

=> Bagaimana bila gunakan age sebagai condition? (menggunakan 2 condition).

=> Bila tidak menggunakan condition:

```
app.get('/', function(req, res, next){
        user.find({}, function(err, foundUser){
                if (foundUser){
                        res.json(foundUser);
                } else {
                        res.json("User tidak ada")
                }
        });
})
```

=> Bila menggunakan dinamis URL:

```
app.get('/:name', function(req, res, next){
        user.find({ name: req.params.name },
function(err, foundUser){
                if (foundUser){
                        res.json(foundUser);
                } else {
                        res.json("User tidak ada")
                }
        });
})
```

Coba pada browser localhost:3000/:Ersya dan :Dona, apa yang terjadi?

Perbedaan find dan findOne

Setelah percobaan diatas anda tahu apa perbedaannya?

> findOne mencari dengan spesifik object.

Find dengan spesifik object (user) juga bisa menggunakan ID, setiap kali membuat user baru mongodb membuat unik ID pada user (object) baru tsb.

```
app.get('/:id', function(req, res, next){
        user.findById({ _id: req.params.id },
function(err, foundUser){
                if (foundUser){
                        res.json(foundUser);
                } else {
                        res.json("User tidak ada")
                }
        });
})
```

coba di browser dengan copas _id user yang sudah ada dari postman.

localhost:3000/paste_id_disini

Mongoose custom method

Custom method kita gunakan untuk menambahkan method pada Schema yang kita inginkan (secara spesifik), sebagai contoh kita membuat schema baru sebagai custom function (tuliskan dibawah UserSchema):

```
userSchema.methods.addLastName = function(lastName){
        this.name = this.name + " " + lastName;
        return this.name
}
```

Kemudian update code diatas menjadi:

```
app.get('/:id', function(req, res, next){
        user.findById({ _id: req.params.id },
function(err, foundUser){
                foundUser.addLastName("Bumi");
                foundUser.save(function(err){
                        res.json(foundUser)
                });
        });
})
```

-> Coba di browser lagi seperti diatas dengan copas ID dari Postman.

-> Dan juga cek pada mLab database apakah sudah terupdate.

Keterangan Code

```
userSchema.methods.addLastName = function(lastName){
        this.name = this.name + " " + lastName;
        return this.name
}
```

Code diatas adalah Custome Schema atau sebenarnya ini adalah function yang kita custome untuk menambahkan lastName pada data yang sudah tersimpan didalam database.

```
app.get('/:id', function(req, res, next){
        user.findById({ _id: req.params.id },
function(err, foundUser){
                foundUser.addLastName("Bumi");
                foundUser.save(function(err){
                        res.json(foundUser)
```

```
                    });
            });
    })
```

Code diatas adalah kita menggunakan method GET untuk mencari object berdasarkan ID dan apabila data telah ditemukan ditambahkanlah lastname yang kita inisialkan, kemudian di simpan dan di kembalikan dengan respon json.

Oke Continue to next video 😃
