

Web Tasarımı ve Programlama

Ders 9:Mongoose ile Veritabanı İşlemleri

Dr. Öğr. Üyesi Elvan DUMAN

eduman@mehmetakif.edu.tr
Yazılım Mühendisliği Bölümü

1. Mongoose'a giriş ve ODM mantığı.
2. Bağlantı ve kurulum.
3. Ürün şeması ve model.
4. Mongoose ile CRUD:
 - Kaydetme,
 - Tüm kayıtları getirme
 - Tek kayıt getirme
 - Güncelleme
 - Silme işlemleri

İlk aşamada Node.js uygulamasında **MongoDB (NoSQL)** kullandık. CRUD işlemlerinde **sorguları (query) manuel yazmamız gerekmektedir. Bu durumda;**

- Sorgu yazma yükü artar
- Odak “veri” yerine “query”ye kayar
- Kod tekrarları ve bakım maliyeti yükselir

Bu derste yapacağımız: JavaScript **objelerini** MongoDB'deki **collection içindeki document** yapısına eşlemek.

Projeye **Mongoose** eklemek

MongoDB ile daha düzenli ve yönetilebilir şekilde çalışmak

Hedef: “Sorguları tek tek yaz” yaklaşımından “veriyi tanımla ve onunla çalış” yaklaşımına geçmek

Mongoose = ODM (Object Document Mapping) kütüphanesi

MongoDB **document** tabanlıdır.

- Uygulamadaki **JavaScript objelerini**, MongoDB'deki **collection** içindeki **document** yapısına eşler.

Amaç:

- Biz **veri yapısına** odaklanıyoruz.
- Mongoose birçok sorguyu arka planda yönetir.
- Yine de gerektiğinde sorguları özelleştirebiliriz.

Temel kavramlar:

- **Schema:** Verinin yapısı (alanlar, tipler, kurallar)
- **Model:** Schema'nın veritabanı ile etkileşen hali
- **Instance:** Model'den üretilen gerçek nesne (kaydetme/sorgulama işlemlerinde kullanılır)

- Resmî dokümantasyon: <https://mongoosejs.com/>



- Bu ders bir “Mongoose dersi” değildir, amaç temel kullanımı öğretmektir.
- Mongoose’u projeye eklemek için paket kurulumu yapılır.
npm install --save mongoose
- Sonraki adım, uygulamayı veritabanına bağlamaktır. Mongoose bağlantı yönetimini arka planda kendisi yapar. Bu yüzden ayrı bir database.js yardımcı dosyasına ihtiyaç kalmaz.
- Bağlantı mongoose.connect(connectionString) ile kurulur.
- connect bir **promise** döndürür, başarıda server başlatılır, hatada hata yakalanır.

App.js mongoose bağlantı örneği

```
// app.js
const express = require("express");
const mongoose = require("mongoose");

const app = express();

mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    console.log("MongoDB bağlantısı kuruldu.");
    app.listen(3000, () => console.log("Server 3000 portunda."));
  })
  .catch((err) => {
    console.log("MongoDB bağlantı hatası:", err);
  });
```

- Mongoose ile verinin yapısını **Schema** ile tanımlarız.
- MongoDB esnektir; ancak uygulamada tutarlı veri yapısı için alanları tanımlamak gerekir.
- Alanlara **type** verilir ve gerekiyorsa **required: true** ile zorunlu hale getirilir.
- `_id` alanını yazmayız; MongoDB otomatik üretir.
- Modeller genellikle `models` klasörü içinde saklanır.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const urunSchema = new Schema({
  urun_adi: {
    type: String,
    required: true
  },
  fiyat: {
    type: Number,
    required: true
  },
  aciklama: {
    type: String,
    required: true
  }
});
module.exports = mongoose.model('Urun', urunSchema);
```

- Controller, formdan gelen verileri req.body ile alır.
- Mongoose'ta yeni kayıt oluşturmak için **Modelden yeni bir instance** üretilir.
- Alan adları, **senin şemadaki alan adlarıyla aynı** olmalıdır: urun_adi, fiyat, aciklama.
- save() metodu Mongoose tarafından gelir ve document'i MongoDB'ye ekler.
- Kayıt başarılı olursa anasayfaya yönlendirilir.

```
exports.post_urun_ekle = (req, res, next) => {  
  const urun_adi = req.body.urun_adi;  
  const fiyat = req.body.fiyat;  
  const aciklama = req.body.aciklama;  
  
  const urun = new Urun({  
    urun_adi: urun_adi,  
    fiyat: fiyat,  
    aciklama: aciklama  
  });  
  urun.save()  
    .then(result => {  
      console.log(result);  
      res.redirect('/');  
    })  
    .catch(err => {  
      console.log(err);  
    });  
};
```

- Artık ürün kaydedebildiğimiz için, listelemek için ürünleri çekmemiz gerekir.
- Mongoose'ta bizim yazdığımız fetchAll() gibi bir metoda ihtiyaç yoktur.
- Bunun yerine model üzerinde hazır gelen statik metodlardan find() kullanılır.
- find() Mongoose ile kullanıldığında cursor döndürmek zorunda değildir, doğrudan kayıtları bir dizi olarak getirir.
- Çok büyük veri bekleniyorsa cursor veya limit gibi yöntemlerle veri miktarı sınırlandırılabilir.

```
exports.get_urunler = (req, res, next) => {
  Urun.find()
    .then(urunler => {
      res.render('index', {
        urunler: urunler,
        pageTitle: 'Anasayfa',
      });
    })
    .catch(err => {
      console.log(err);
    });
};
```

- Tüm ürünleri listeledikten sonra, “detay” sayfası için tek bir ürünü çekmemiz gerekir.
- Controller’da URL’den gelen `urun_id` zaten string olarak gelir.
- Mongoose modelinde hazır gelen `findById()` metodu kullanılır.
- Ekstra dönüşüm yapmaya gerek yoktur: Mongoose string ID’yi otomatik olarak `ObjectId`’ye çevirir.
- Sonuç olarak gelen tek ürün nesnesi view’a gönderilir ve detay sayfası render edilir.

```
exports.get_urun = (req, res, next) => {  
  const urun_id = req.params.urun_id;  
  
  Urun.findById(urun_id)  
    .then(urun => {  
      res.render('urun_detay', {  
        urun: urun,  
        pageTitle: 'Ürün Detayı'  
      });  
    })  
    .catch(err => {  
      console.log(err);  
    });  
};
```

- “Düzenle” sayfası açılırken ilgili ürün, `urun_id` ile veritabanından çekilir.
- Form, ürünün mevcut değerleriyle doldurulur.
- “Güncelle” butonuna basılınca yeni bir ürün oluşturulmaz.
- Mevcut ürün tekrar `findById()` ile bulunur, alanları güncellenir ve aynı doküman üzerinde `save()` çağrılır.
- Mongoose, mevcut dokümanda `save()` çağrıldığında ekleme değil, güncelleme işlemi yapar.

```
exports.get_urun_duzenle = (req, res, next) => {
  const urun_id = req.params.urun_id;

  Urun.findById(urun_id)
    .then(urun => {
      if (!urun) {
        return res.redirect('/');
      }
      res.render('urun-duzenle', {
        pageTitle: 'Ürün Düzenle',
        urun: urun,
        yetkili_mi: req.session.yetkili_mi
      });
    })
    .catch(err => console.log(err));
};
```

```
exports.post_urun_duzenle = (req, res, next) => {
  const urun_id = req.body.urun_id;
  const urun_adi = req.body.urun_adi;
  const fiyat = req.body.fiyat;
  const aciklama = req.body.aciklama;

  Urun.findById(urun_id)
    .then(urun => {
      urun.urun_adi = urun_adi;
      urun.fiyat = fiyat;
      urun.aciklama = aciklama;
      return urun.save();
    })
    .then(() => {
      console.log('Ürün güncellendi!');
      res.redirect('/');
    })
    .catch(err => console.log(err));
};
```

- CRUD'in son adımı olarak ürün silmeyi de Mongoose ile yaparız.
- Silme işlemi, controller'daki "post delete" aksiyonunda tetiklenir.
- Mongoose, silme için hazır metotlar sunar.
- Route aktif edilince "Sil" butonuna basıldığında ilgili ürün veritabanından kaldırılır.
- Silme sonrası anasayfaya yönlendirme yapılır.

```
exports.post_urun_sil = (req, res, next) => {  
  const urun_id = req.body.urun_id;  
  
  Urun.findByIdAndDelete(urun_id)  
    .then(() => {  
      console.log('ürün silindi!');  
      res.redirect('/');  
    })  
    .catch(err => {  
      console.log(err);  
    });  
};
```

Ders 9 Sunum Sonu

İletişim : eduman@mehmetakif.edu.tr

