



Node.js

Yönetim Paneli

# Yönetici Paneli

```
routes
JS admin.js
JS user.js
views
  admin
    categoryCreator.ejs
    categoryList.ejs
    movieCreator.ejs
    movieList.ejs
  partials
    categoryMenu.ejs
    head.ejs
    movieCards.ejs
    nav.ejs
    noProducts.ejs
```

```
44
45 router.get("/categories", (req, res) => {
46   res.render('admin/categoryList', {
47     title: "Kategori Listesi",
48   })
49
50 router.get("/movies", (req, res) => {
51   res.render('admin/movieList', {
52     title: "Film Listesi",
53   })
54 module.exports = router;
```

Admin rotasyonunda kullanacağımız admin EJS dosyalarını **views/admin** dizinine ekliyoruz ve bu dosyaları **admin.js** dosyasında render ediyoruz.

MySQL veritabanındaki bilgileri kullanarak bir kategori kataloğu oluşturmak için, **ejsout** ve bir dizi kullanarak **ejs.each** metodunu kullanıyoruz. Bu yöntemle, veritabanından kategori başlıklarını çekiyoruz ve EJS döngüsüyle her bir kategoriye görüntüleyebiliriz.

```
class="container mt-3">
  class="row">
    <div class="col">
      <form action="" method="post">
        <div class="row mb-3 mt-4">
          <label for="baslik" class="col-sm-2 col-form-label">Başlık:</label>
          <div class="col-sm-8">
            <input type="text" class="form-control" id="baslik">
          </div>
        </div>
        <div class="row mb-3 mt-4">
          <label for="aciklama" class="col-sm-2 col-form-label">Açıklama:</label>
          <div class="col-sm-8">
            <textarea class="form-control" id="aciklama">
          </div>
        </div>
        <div class="row mb-3 mt-4">
          <label for="resim" class="col-sm-2 col-form-label">Resim:</label>
          <div class="col-sm-8">
            <input type="text" class="form-control" id="resim">
          </div>
        </div>
        <div class="row mb-3 mt-4">
          <label for="kategori" class="col-sm-2 col-form-label">Kategori:</label>
          <div class="col-sm-8">
            <select class="form-select" id="kategori">
              <option value="" selected>Seçiniz</option>
              <option value="1">Kategori 1</option>
              <option value="2">Kategori 2</option>
            </select>
          </div>
        </div>
        <div class="row mb-3 mt-4">
          <div class="col-sm-8">
            <button type="button" class="btn btn-secondary">Arasayfa</button>
            <button type="button" class="btn btn-primary">Kaydet</button>
          </div>
        </div>
      </form>
    </div>
  </div>
```

Formun methodunu "post" olarak belirtmeyi kesinlikle unutmamamız gerekiyor. Bu, form verilerinin güvenli bir şekilde sunucuya iletilmesini sağlar.

Sonra formumuzu oluşturuyoruz çünkü artık MySQL değil site üzerinden veri girebilmemiz gerek.

```
<div class="col-sm-8">
  <select name="kategori" id="kategori" name="kategori" class="form-select">
    <option value="" selected>Seçiniz</option>
    <% categories.forEach(category => { %>
      <option value="<%= category.categoryid %>"><%= category.categoryName%></option>
    <% }) %>
  </select>
</div>
```

```

    title: kategoriEkle, })
  })
}

router.get("/moviesCreate", async(req,res)=>{
  try {
    const [category,] = await db.execute("select* from category")
    res.render('admin/movieCreator',{
      title: "Film Ekle",
      categories: category
    })
  } catch (error) {
    console.log(error)
  }
})

```

```

router.post("/moviesCreate", async(req,res)=>{
  const baslik = req.body.baslik;
  const aciklama = req.body.aciklama;
  const resim = req.body.resim;
  const kategori = req.body.kategori;
  const anasayfa = req.body.anasayfa == "on" ? 1:0;
  const onay = req.body.onay == "on" ? 1:0;

```

```

  try {
    await db.execute("INSERT INTO movie(movieTitle, description, image, main, onay, categoryid) VALUE(?,?,?,?,?,?)",
    [baslik, aciklama, resim, kategori, anasayfa, onay]);
    res.redirect("/")
  } catch (error) {
    console.log(error);
  }
});

```

```

router.get("/movies/:movieid", (req,res)=>{
  res.render('admin/movieEdit')
})

```

```
const db = require("../data/db")
```

Aynı zamanda, admin.js içinde olması gereken bilgi, çünkü db.js, bağlantı config oluşturduğumuz yerdir.

index.js içerisine !!!

```
app.use(express.urlencoded({extended: false}));
```

Sırası yanlış en sona alınmalı yukarıdaki gibi

Bu komut middleware'i, gelen isteklerdeki URL kodlamasını ve form verilerini işlemek için kullanılır. Özellikle, bu middleware form verilerini **req.body** üzerinde erişilebilir hale getirir.

Yeni eklediğimiz **movieList.ejs** dosyasına bir Bootstrap tablosu ekleyerek, MySQL'den alınan veriyi tabloya aktarıyoruz.

```
> admin > movieList.ejs > html > body > ? > div.container > div.row > div.container > table > table-striped > table
<html lang="en">
<body>
  <%- include('../partials/nav') %>
  <div class="container mt-3">
    <div class="row">
      <table class="table table-striped">
        <thead>
          <tr>
            <th>id</th>
            <th>Resim</th>
            <th>Başlık</th>
            <th></th>
          </tr>
        </thead>
        <tbody>
          <% movies.forEach(e => { %>
            <tr>
              <td><%= e.movieid %></td>
              <td></td>
              <td><a style="text-decoration: none;" class="text-dark" href="/movies/<%= e.movieid %>"> <%= e.movieTitle %></a></td>
              <td><a class="btn btn-sm btn-primary" href="/movies/<%= e.movieid %>">Düzenle</a>
                <a class="ms-2 btn btn-sm btn-danger" href="/movies/<%= e.movieid %>">Sil</a></td>
            </tr>
          <% }) %>
        </tbody>
      </table>
    </div>
  </div>
</body>
</html>
```

**Admin.js'e girip router'ımızı MySQL'den filmleri çekecek şekilde ayarlıyoruz. Artık her şey daha kısa ve kolay hale gelmeye başladı.**

```
49
50
51
52
53
54
55
56
57
58
59
60
61
router.get("/movies", async(req, res) => {
  try {
    const [movies, ] = await db.execute("select movieid, movieTitle, image from movie")
    res.render('admin/movieList', {
      title: "Film Listesi",
      movies: movies
    });
  } catch (error) {
    console.log(error);
  }
})
module.exports = router;
```





# Güncelleme işlemi

```
> adminjs > ...

router.get("/movies/:movieid", async(req, res) => {
  const movieid = req.params.movieid;
  try {
    const [movies, ] = await db.execute('select * from movie where movieid=?', [movieid])
    const [category, ] = await db.execute('select * from category')
    const movie = movies[0]
    if(movie){
      res.render('admin/movieEdit', {
        title: movie.movieTitle,
        movie: movie,
        categories: category
      })
    }
  } catch (error) {
    console.log(error)
  }
})

router.post("/movies/:movieid", async(req, res) => {
  const movieid = req.body.movieid;
  const baslik = req.body.baslik;
  const aciklama = req.body.aciklama;
  const resim = req.body.resim;
  const anasayfa = req.body.anasayfa == "on" ? 1:0;
  const onay = req.body.onay == "on" ? 1:0;
  const kategori = req.body.kategori;
  try {
    await db.execute('UPDATE movie SET movieTitle=?, description=?, image=?, main=?, onay=?, categoryid=? WHERE movieid=?', [baslik, aciklama,
    resim, anasayfa, onay, kategori, movieid]);
    res.redirect("/admin/movies")
  } catch (error) {
    console.log(error);
  }
})

router.get("/movies/:movieid", async(req, res) => {
```

İşte gizli input verisindeki movieid;

`<input type="hidden" name="movieid" value="<%= movieid %>">`

**Güncelleme işlemi için MySQL'de kullanım farklı olacaktır. Bir önceki slaytta insert şeklinde kullanmıştık bu sefer güncellediğimiz için update kullanıyoruz.**

**Bu örnek içerisinde, UPDATE sorgusuyla belirli bir film kaydının bilgilerini güncellemeyi sağlayan MySQL sorgusu verilmiştir.**

# Silme işlemi

```
Go Run Terminal Help
categoryCreate.ejs movieCreate.ejs movieDelete.ejs this.ejs movieList.ejs admin.ejs movieDelete.ejs
> admin > movieDelete.ejs > html > body > ? > div.container mt-3 > div.row > div.col-10 offset-1 > div.alert alert-warning row > form > input#movieid
<!DOCTYPE html>
<html lang="en">
<head>
<%- include('../partials/head') %>
</head>
<body>
<%- include('../partials/nav') %>

<div class="container mt-3">
  <div class="row">
    <div class="col-10 offset-1">
      <div class="alert alert-warning row">
        <p class="mb-3">
          <b><%= movie.movieTitle %> </b> <p>isimli film kaydını silmek istiyor musunuz?</p>
        </p>
        <form method="post">
          <input type="hidden" name="movieid" id="movieid" value="<%= movie.movieid %>" />
          <a href="/admin/movieList" class="btn btn-primary ms-2 float-end">İptal</a>
          <button type="submit" class="btn btn-danger float-end">Evet</button>
        </form>
      </div>
    </div>
  </div>
</div>
</body>
```

- **MovieDelete.ejs** dosyasını oluşturduktan sonra, film bilgisine erişmek için gizli bir input içinde film id'sini **ejsout** komutuyla sisteme iletiyoruz.
- **Router.post** ile movieid'ye req.body.movieid değerini atıyoruz. Böylelikle DELETE FROM komutu ile veriyi silebiliyoruz. Tabii ki, önceki movieList'teki sil butonunun a href'ini düzenleyerek bu dosyaya aktarıyoruz.

- Delete işlemi için kullanıcıya emin olup olmadığını sormamız gereken bir yönlendirme yapmalıyız.
- EJS Dosyası Oluşturma: "movieDelete.ejs" adında bir EJS dosyası oluşturun.
- Rota Bağlantısı: `app.get('/delete/:movieId', ...)` rotasında, "movieDelete.ejs" dosyasını render etmek için rota ile bağlantı oluşturun.

```
6
7
8 router.get("/movieDelete/:movieid", async(req,res)=>{
9   const movieid = req.params.movieid;
10  try {
11    const [movies, ] = await db.execute('select * from movie where movieid=?',[movieid]);
12    const movie = movies[0];
13    res.render('admin/movieDelete',{
14      title: "Film Silme",
15      movie: movie
16    })
17  } catch (error) {
18    console.log(error)
19  }
20 })
21
22 router.post("/movieDelete/:movieid", async(req,res)=>{
23   movieid = req.body.movieid
24   try {
25     await db.execute('DELETE from movie where movieid=?',[movieid])
26     res.redirect("/admin/movies")
27   } catch (error) {
28     console.log(error)
29   }
30 })
```



## MovieList.ejs içine ejsif bloğunda;

```
<% include('../partials/nav') %>
<div class="container mt-3">
  <div class="row">
    <div class="col-10 offset-1">
      <% if (action!="undefined" && action=="update") { %>
        <div class="alert alert-primary alert-dismissible" role="alert">
          <strong></strong> Film güncellendi!
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Kapat"></button>
        </div>
      <% } %>
      <% if (action!="undefined" && action=="create") { %>
        <div class="alert alert-success alert-dismissible" role="alert">
          <strong></strong> Film eklendi!
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Kapat"></button>
        </div>
      <% } %>
      <% if (action!="undefined" && action=="delete") { %>
        <div class="alert alert-danger alert-dismissible" role="alert">
          <strong></strong> Film silindi!
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Kapat"></button>
        </div>
      <% } %>
      <table class="table table-striped">
        <tbody>
          <tr>
            <td></td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

bunun sebebi iste url de bu kodların yazması  
<http://127.0.0.1:8080/admin/movies?action=update> veya delete ya da create

Şimdi farklı kaynaklardan ve yaptıklarınızdan yola çıkarak kategori listesi ve kategori ekleme sayfası yapın --->

`res.redirect()` yöntemi kullanılarak yönlendirme yapılırken, isteğin yönlendirildiği URL'ye isteğe bağlı olarak bir sorgu dizesi (query string) ekleyebilirsiniz. İşte `res.redirect()` kullanırken bu işlemi gerçekleştirmenin bir örneği:

```
// Güncelleme işlemi tamamlandığında
res.redirect("/admin/movies?action=update");
```

```
// Silme işlemi tamamlandığında
res.redirect("/admin/movies?action=delete");
```

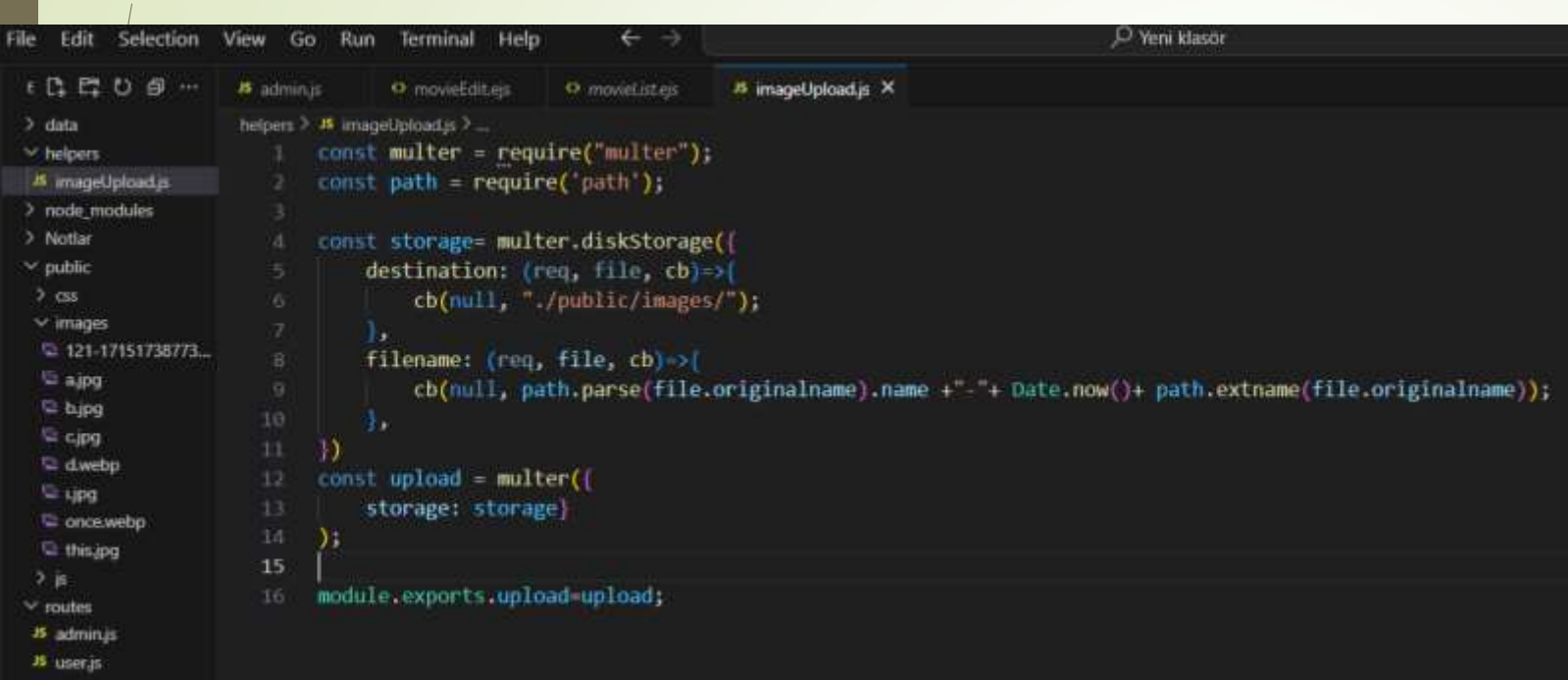
```
// Oluşturma işlemi tamamlandığında
res.redirect("/admin/movies?action=create");
```

`action: req.query.action`


isteğin yönlendirildiği sayfada `action` parametresini almak için bir route oluşturmanız gerekiyor



Formumuzun **enctype** özelliğini **multipart/form-data** olarak değiştirmemiz gerekiyor. Bunu yapmamızın sebebi, resimlerin artık dosya yolu yerine dosya yükleme yoluyla yüklenmesi gerektiğidir. Bu işlemi gerçekleştirebilmek için **npm i multer** komutunu terminale girip Multer paketini indirmeniz gerekmektedir.



```
File Edit Selection View Go Run Terminal Help
helpers > JS imageUpload.js > ...
1 const multer = require("multer");
2 const path = require('path');
3
4 const storage= multer.diskStorage({
5   destination: (req, file, cb)=>{
6     cb(null, "../public/images/");
7   },
8   filename: (req, file, cb)=>{
9     cb(null, path.parse(file.originalname).name + "-" + Date.now() + path.extname(file.originalname));
10  },
11 });
12 const upload = multer({
13   storage: storage
14 });
15
16 module.exports.upload=upload;
```



```
<div class="row">
  <div class="col">
    <form action="" method="post" enctype="multipart/form-data">
      <input type="hidden" name="movieId" value="{< movie.movieId}<}">
      <input type="hidden" name="resim" value="{< movie.image}<}">
    </div class="row mb-3 mt-4">
      <label for="baslik" class="col-sm-2 col-form-label">Başlık:</label>
```

Dosya dizinimize, multer için bir klasör olan 'imageUpload'ı yeni olarak oluşturuyoruz.

Artık admin.js te tek tek yazabiliriz.

## Admin.js dosyasına, başlangıçta

'const imageUpload = require("../helpers/imageUpload")'

kodunu ekleyin. Ardından, imageUpload'ı kullanmaya başlayabiliriz.

```
92 }
93 })
94
95
96
97
98 router.post("/moviesCreate", imageUpload.upload.single("resim"), async(req,res)=>{
99   const baslik = req.body.baslik;
100   const aciklama = req.body.aciklama;
101   const resim = req.file.filename;
102   const anasayfa = req.body.anasayfa == "on" ? 1:0;
103   const onay = req.body.onay == "on" ? 1:0;
104   const kategori = req.body.kategori;
105
106
107   try {
108     await db.execute("INSERT INTO movie(movieTitle, description, image, main, onay, category,
109     anasayfa, onay, kategori)");
110     res.redirect("/admin/movies?action=create")
111   } catch (error) {
112     console.log(error);
113   }
114 });
```

- Birkaç değişiklik yaparak işimizi halledebiliriz.
- MovieCreate.ejs ve movieEdit.ejs dosyalarında, artık dosya yükleyeceğimiz için inputların typelarını file yapmalıyız. Daha sonra, güncelleme için kullanıcının eski dosyaları görmesini istiyoruz. Bu amaçla, eski resmi bu şekilde gösterebiliriz.
- Aynı şekilde, create içinde de aynısını yapabiliriz, gerek yok.

```
</div>
<div class="row mb-3 mt-4">
  <label for="resim" class="col-sm-2 col-form-label">Resim</label>
  <div class="col-sm-8">
    
    <input type="file" class="form-control" id="resim" name="resim">
  </div>
</div>
```

```
Go Run Terminal Help ← → Yeni klasör

min.js × < movieEdit.ejs < movieCreator.ejs JS imageUpload.js

s > JS admin.js > router.post("/moviesCreate") callback
})

router.post("/movies/:movieid", imageUpload.upload.single("resim"), async(req,res)=>{
  const movieid = req.body.movieid;
  const baslik = req.body.baslik;
  const aciklama = req.body.aciklama;
  let resim = req.body.resim;

  if(req.file){
    resim= req.file.filename;
    fs.unlink("./public/images/"+req.body.resim, err=>{
      console.log(err)
    })
  }

  const anasayfa = req.body.anasayfa == "on" ? 1:0;
  const onay = req.body.onay == "on" ? 1:0;
  const kategori = req.body.kategori;
  try {
    await db.execute("UPDATE movie SET movieTitle=?, description=?, image=?, main=?, onay=?, baslik=?, aciklama=?, kategori=?, resim=? WHERE movieid=?");
    res.redirect("/admin/movies?action=update")
  }catch (error) {
    console.log(error);
  }
})
```

- Güncelleme sayfasında eski resmin silinmesi için Node.js'te fs modülünü kullanabiliriz. İlk olarak, fs modülünü require ediyoruz ve sonra fs.unlink metodunu kullanarak movieEdit sayfasından resim ID'sini alıp silebiliriz.
- Bu fonksiyonu, kullanıcının eski resmi silmek istediği zaman movieEdit sayfasında çağırabiliriz. Örneğin, bir "Sil" butonuna tıkladığında veya bir form gönderildiğinde çağırılabilir. Bu, eski resmin silinmesini sağlar.
- Özetlemek gerekirse, fs modülünü kullanarak eski resmi silmek için önce dosya yolunu belirleyip, sonra fs.unlink metodunu kullanarak dosyayı silmeliyiz. Bu sayede güncelleme işlemi sırasında eski resimlerin temizlenmesini sağlayabiliriz.