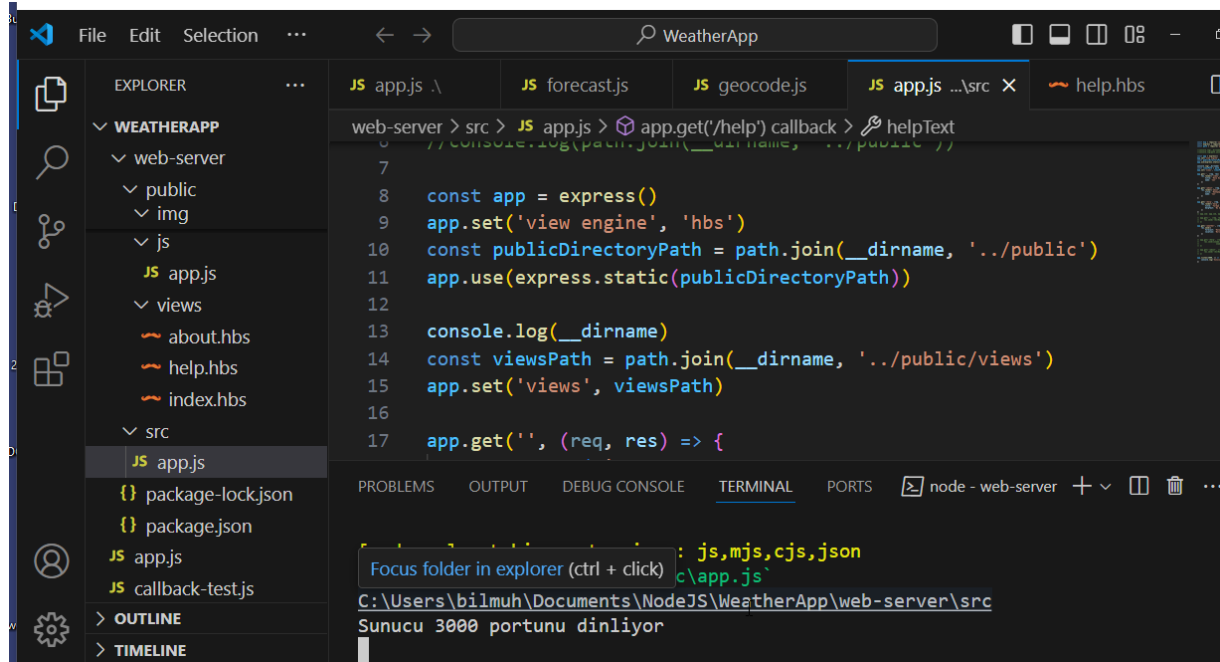


BURSA TEKNİK ÜNİVERSİTESİ NODE.JS ile WEB PROGLAMA DERSİ 8.HAFTA

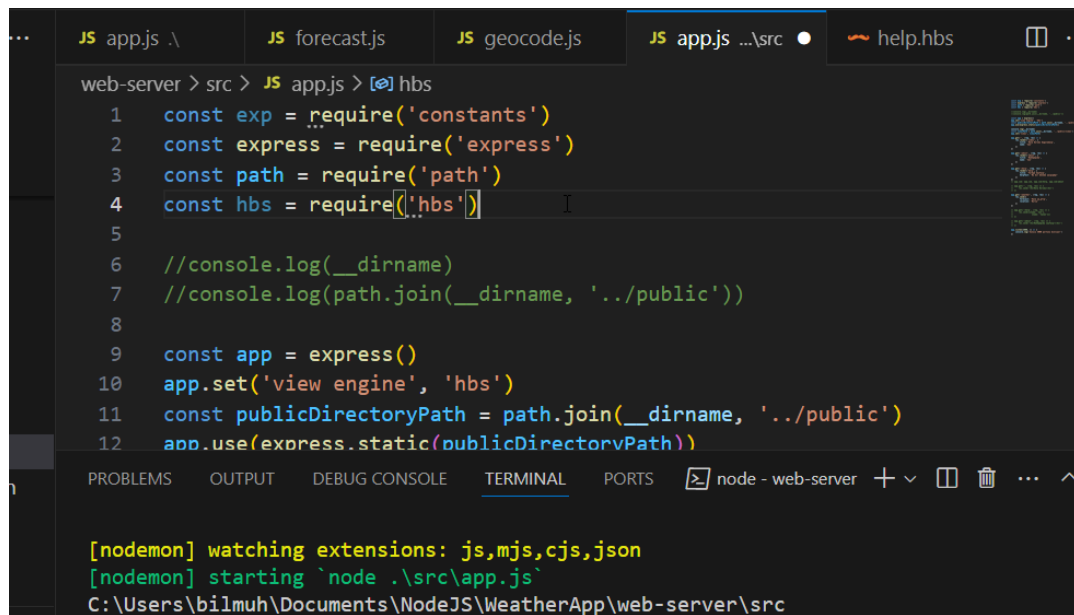
TEORİ DERSİ RAPORU

Web sayfalarında tekrar eden kısımları (header, footer) bulunur. Bu tekrar eden yapılarda değişiklik yaptığınızda tüm sayfalarda uygulanmasını isteriz. Her sayfada bu değişiklikleri teker teker yapmak zor olacaktır. Daha kolay hale getirmek için partial yöntemi kullanılarak sayfalar içinde tekrar eden kısımları daha kullanışlı hale getirilir. Uygulamamıza başlamadan önce geçen haftaki kodumuzu çalıştıralım ve tarayıcı da kontrol edelim sayfamız erişilebilir olması gereklidir. Geçen haftaki kodumuz:



```
web-server > src > JS app.js > app.get('/help') callback > helpText
7
8 const app = express()
9 app.set('view engine', 'hbs')
10 const publicDirectoryPath = path.join(__dirname, '../public')
11 app.use(express.static(publicDirectoryPath))
12
13 console.log(__dirname)
14 const viewsPath = path.join(__dirname, '../public/views')
15 app.set('views', viewsPath)
16
17 app.get('/', (req, res) => {
```

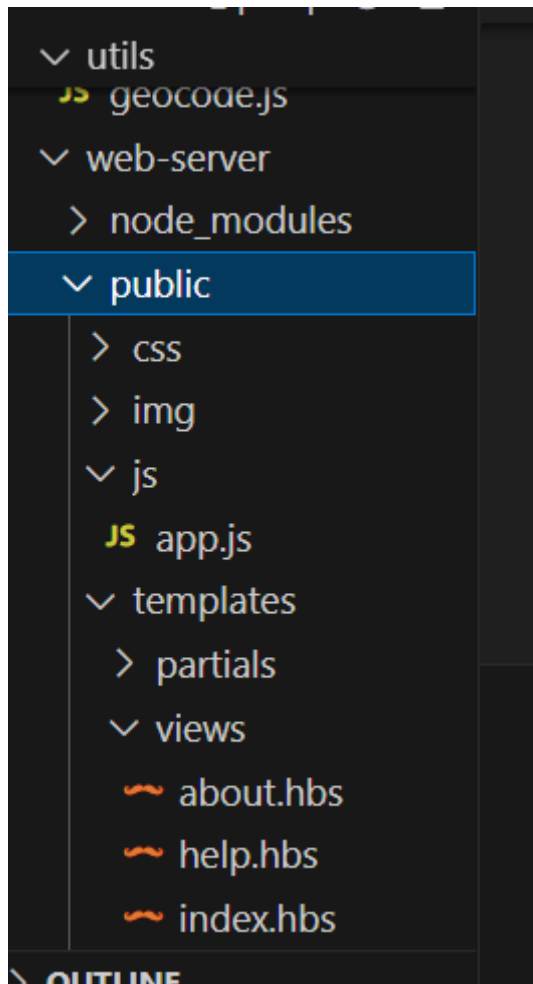
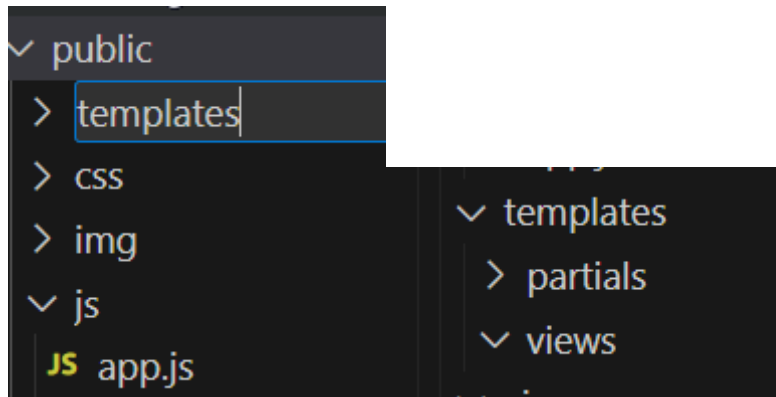
Hbs modülünü kullanacağız ve dahil etme işlemini yapalım.



```
web-server > src > JS app.js > hbs
1 const exp = require('constants')
2 const express = require('express')
3 const path = require('path')
4 const hbs = require('hbs')
5
6 //console.log(__dirname)
7 //console.log(path.join(__dirname, '../public'))
8
9 const app = express()
10 app.set('view engine', 'hbs')
11 const publicDirectoryPath = path.join(__dirname, '../public')
12 app.use(express.static(publicDirectoryPath))
```

```
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node .\src\app.js`
C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\src
```

Const hbs=require("hbs"); kodu yazılarak hbs modülünü dahil ederiz. Public klasörü altında templates isimli bir klasör açalım. Views klasörünü templates klasörünün altına bırakalım. Daha önce oluşturduğumuz views klasörünü içinde bulunan dosyaları yeni views içine taşıyalım ve templates klasörünün altına partials isimli bir klasör daha oluşturalım.



Daha sonra app.js içinde viewsPath kısmında konum bilgisi güncellemesi yapalım artık public altında templates onun da altında views klasörü bulunmaktadır.

```

8
9  const app = express()
10 app.set('view engine', 'hbs')
11 const publicDirectoryPath = path.join(__dirname, '../public')
12 app.use(express.static(publicDirectoryPath))
13
14 console.log(__dirname)
15 const viewsPath = path.join(__dirname, '../public/templates/views')
16 app.set('views', viewsPath)
17
18

```

Konum bilgisi güncellenmiş hali yukarıdaki görseldir. Bu aşamada partialsPath' i oluşturalım.

```

server > src > JS app.js > ...
  app.set('view engine', 'hbs')
  const publicDirectoryPath = path.join(__dirname, '../public')
  app.use(express.static(publicDirectoryPath))

  console.log(__dirname)
  const viewsPath = path.join(__dirname, '../public/templates/views')
  app.set('views', viewsPath)

  const partialsPath = path.join(__dirname, '../public/templates/partia
  app.get('', (req, res) => {
    res.render('index', {

```

Path.join ile bu işlem gerçekleştirilir. Srcden bir üste geçiş yapmamız lazım.

```

web-server > src > JS app.js > [0] partialsPath
10  set('view engine', 'hbs')
11  t publicDirectoryPath = path.join(__dirname, '../public')
12  use(express.static(publicDirectoryPath))
13
14  ole.log(__dirname)
15  t viewsPath = path.join(__dirname, '../public/templates/views')
16  set('views', viewsPath)
17
18  t partialsPath = path.join(__dirname, '../public/templates/partials')
19
20  get('', (req, res) => {
21  res.render('index', {

```

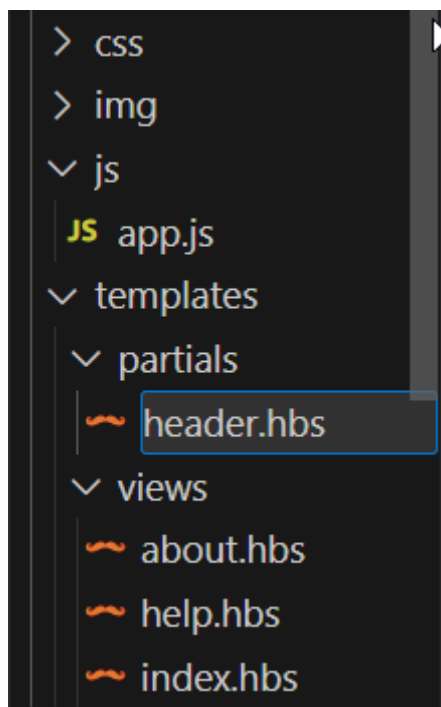
Elde ettiğimiz partialPath bilgisi ile hbs.registerPartials() ile hbs artık partials ifadelerinin nerede bulunduğunu bilecektir.

```

web-server > src > JS app.js > ...
10 app.set('view engine', 'hbs')
11 const publicDirectoryPath = path.join(__dirname, '../public')
12 app.use(express.static(publicDirectoryPath))
13
14 console.log(__dirname)
15 const viewsPath = path.join(__dirname, '../public/templates/views')
16 app.set('views', viewsPath)
17
18 const partialsPath = path.join(__dirname, '../public/templates/partials')
19 hbs.registerPartials(partialsPath)
20
21

```

Partials klasörü altında header.hbs dosyamızı açalım.



Header.hbs içine basit bir html kodu yazalım.

```

web-server > public > templates > partials > header.hbs > h1 > ?
1 <h1>Static Header.hbs</h1>

```

Şu an bu yazı statiktir bu ifadeyi kaydedelim peki bu partials ifadesini nerede kullanacağız? Bu sorunun cevabı için help.hbs içine gidelim. Help.hbs dosyasının ilk hali bu şekilde:

```
forecast.js  JS geocode.js  JS app.js ...src  header.hbs  help
web-server > public > templates > views > help.hbs > html > body

3  <html>
4    <head>
5      <link rel="stylesheet" href="/css/styles.css">
6    </head>
7
8    <body>
9      <h1>{{title}}</h1>
10     <p>{{helpText}}</p>
11   </body>
12 </html>
```

Body içine partials yazımızı ekleyelim. Bunun için `{{>header}}` yazmamız gereklidir daha sonra kaydedelim ama kaydetmemize rağmen nodemon yenilenmiyor bu bir problemdir. Bunun düzeltmenin yolu ise öncelikle nodemonu durdurmamız gerekli daha sonra çalıştırırken nodemon `.\src\app.js -e js, hbs` yazmalıyız.


```
web-server > public > templates > views > help.hbs > html > body

3  <html>
4    <head>
5      <link rel="stylesheet" href="/css/styles.css">
6    </head>
7
8    <body>
9      {{>header}}
10     <h1>{{title}}</h1>
11     <p>{{helpText}}</p>
12   </body>
13 </html>
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  powershell - web-server  + v  [ ]  [X]  ...

[nodemon] starting `node .\src\app.js`
C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\src
Sunucu 3000 portunu dinliyor
PS C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server> nodemon .\src\app.js -e js,hbs
```

Nodemon normalde hbs ve js dosyalarındaki değişiklikleri takip etmiyor ama bu kod ifadesi ile restart olmasına zorlayabiliriz.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  node - web-server

[nodemon] watching extensions: js,hbs
[nodemon] starting `node .\src\app.js`
C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\src
Sunucu 3000 portunu dinliyor
```

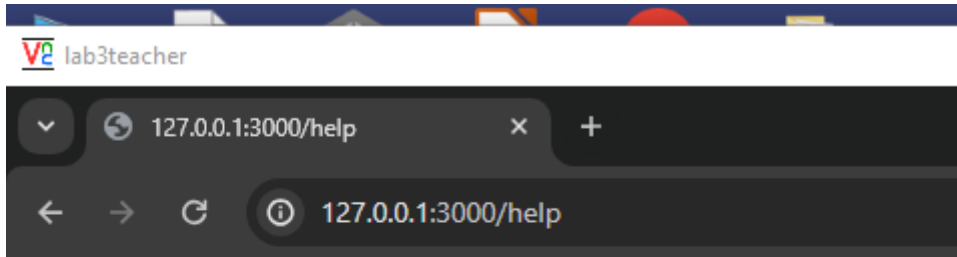
Restart yapmaya zorlamış oluruz hbs de değişiklik gördüğünde de restart yapar. Tarayıcıda help dosyasına gidelim ama bir hata ile karşılaştık.

```
lab3teacher

Error
127.0.0.1:3000/help

Error: Failed to lookup view "help" in views directory "C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\public\views"
    at Function.render (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\application.js:597:17)
    at ServerResponse.render (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\response.js:1048:7)
    at C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\src\app.js:32:9
    at Layer.handle [as handle_request] (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\layer.js:95:5)
    at next (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\route.js:149:13)
    at Route.dispatch (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\route.js:119:3)
    at Layer.handle [as handle_request] (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\layer.js:95:5)
    at C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\index.js:284:15
    at Function.process_params (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\index.js:346:12)
    at next (C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\node_modules\express\lib\router\index.js:280:10)
```

Dosyayı az önce kayıt etmediğimiz için bu hatayı aldık eğer kayıt edersek şu çıktıyı alırız.



Static Header.hbs

Yardım Sayfası

Bu bir deneme yazısıdır

Bu yazı hala static ve farklı bir dosyadan static header.hbs yazısı geliyor amacımız yazıyı dinamik hale getirerek içeriğin parametrik değişmesini sağlayacağız. Buna bağlı olarak bazı değişiklikler yapalım. Help.hbs içindeki title bilgisini silelim bu bilgi artık header.hbs içinden gelecektir ve kayıt edelim index.hbs ye gidelim.

web-server > public > templates > views > help.hbs > html > body

```
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10
11     <p>{{helpText}}</p>
12   </body>
13 </html>
```

Yine buradan title kısmını kaldıralım ve {{>header}} kodunu ekleyelim. Eğer başında > sembolü varsa bu onun partials olduğunu ifade eder.

web-server > public > templates > views > index.hbs > html > body

```
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6     <script src="/js/app.js"></script>
7   </head>
8
9   <body>
10    {{>header}}
11    <p>{{name abc header
12  </body>
13 </html>
```

Başından > varsa partials klasörü içinden veri alınacak anlamına gelir. Name ise app.js içinden gelir.

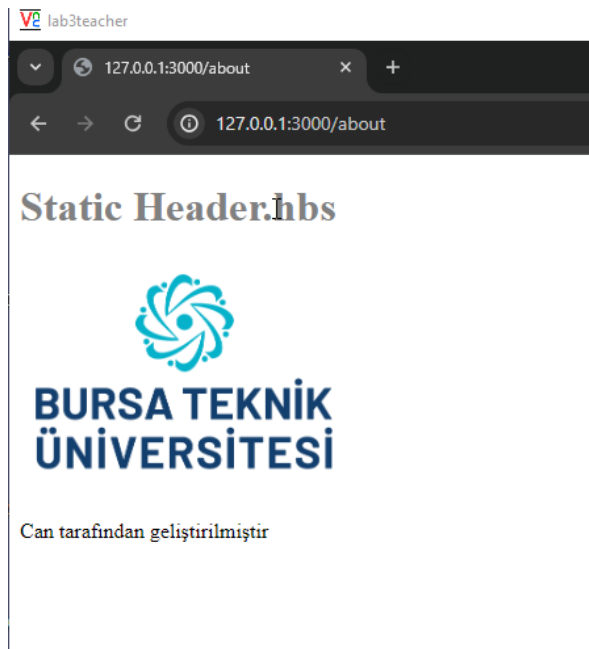
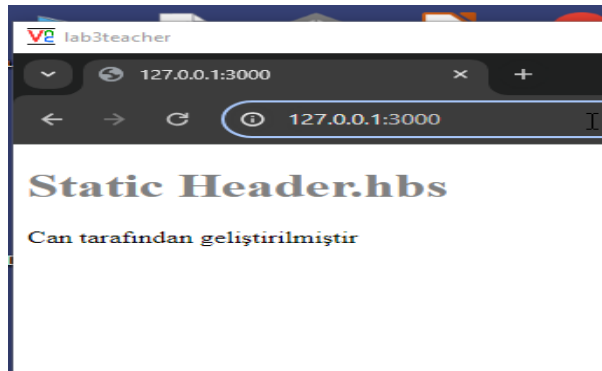
```
web-server > src > JS app.js > ...
18 const partialsPath = path.join(__dirname, '../public/templates/partials');
19 hbs.registerPartials(partialsPath);
20
21 app.get('/', (req, res) => {
22   res.render('index', {
23     title: 'Hava Durumu Uygulaması',
24     name: 'Can'
25   });
26 });
27
28 app.get('/about', (req, res) => {
```

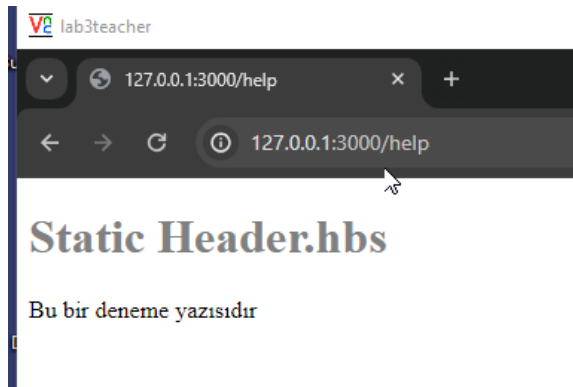
Kaydedelim ve about.hbs içine gidelim. Bu kısımda da {{>header}} kodunu ekleyip header bilgisini silelim.

```
web-server > public > templates > views > about.hbs > html > body
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    <img src=abc header
11    <p>{{name abc Header
12  </body>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS node - web-server
[nodemon] restarting due to changes...
```

Yaptığımız işlemleri kaydedelim ve test işlemi yapalım. Tarayıcı da sayfalarımıza bakalım.





Help.hbs içinde yazan kodlarımız şu şekildeydi:

```
JS app.js ...src header.hbs help.hbs index.hbs about.hbs
web-server > public > templates > views > help.hbs > html > body > p
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    <p>{{helpText}}</p>
11  </body>
12 </html>
```

App.js içine gidelim eskiden yazdığımız kodlar:

```
JS app.js ...src X header.hbs help.hbs index.hbs abo
web-server > src > JS app.js > app.get('/help') callback > helpText
28 app.get('/about', (req, res) => {
33 })
34
35 app.get('/help', (req, res) => {
36   res.render('help', {
37     title: 'Yardım Sayfası',
38     helpText: 'Bu bir deneme yazısıdır'
39   })
40 })
41 // app.com: app.com, app.com/help, app.com/about
42
43 // app.get('', (req, res) => {
```

Title ve helpTest ifadelerimiz bulunuyor bunlara ek olarak name ekleyelim ve kaydedelim ve header.hbs içine gidelim.

```
web-server > src > JS app.js > app.get('/help') callback
28 app.get('/about', (req, res) => {
33 })
34
35 app.get('/help', (req, res) => {
36   res.render('help', {
37     title: 'Yardım Sayfası',
38     name: 'Can Deniz',
39     helpText: 'Bu bir deneme yazısıdır'
40   })
41 })
42 // app.com: app.com, app.com/help, app.com/about
```

Buradaki static.headerhbs yazısını parametrik hale getirelim bunun için yazıyı silip {{title}} ifadesini ekleyelim. Help, about ve ana sayfadaki bulunan özel title ifadeleri ekrana basılır hepsinde artık staticheader.hbs yazısı yazmayacaktır.

```
JS app.js ...src header.hbs X help.hbs index.h
web-server > public > templates > partials > header.hbs > h1
1 <h1>{{title}}</h1>
```

3 sayfamızda title bilgimiz var about ve help sayfasında title verisinin bulunduğunu görelim.

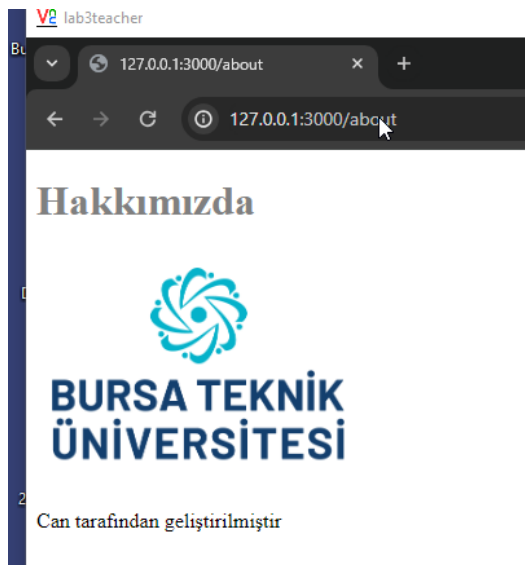
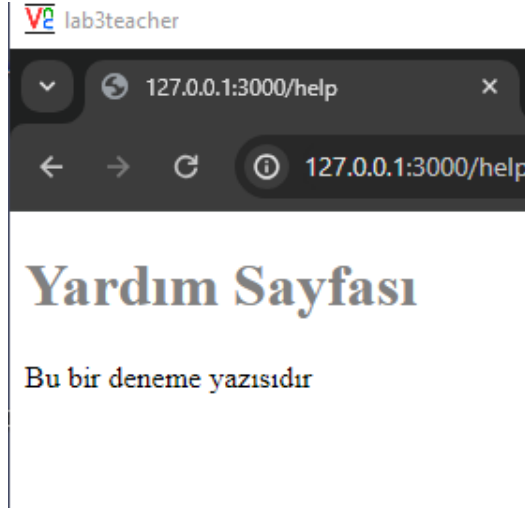
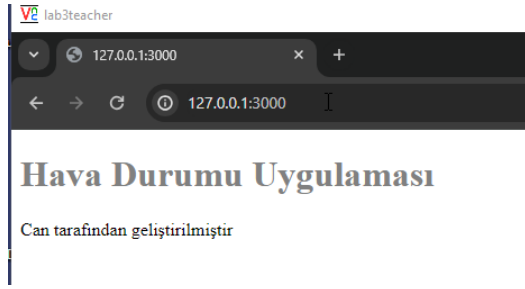
```
server > src > JS app.js > ...
app.get('/about', (req, res) => {
  res.render('about', {
    title: 'Hakkımızda',
    name: 'Can'
  })
})

app.get('/help', (req, res) => {
  res.render('help', {
    title: 'Yardım Sayfası',
    name: 'Can Deniz',
    helpText: 'Bu bir deneme yazısıdır'
  })
})
```

BLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS node - web-server +

```
demon] restarting due to changes...
demon] starting `node .\src\app.js`
Users\bilmut\Documents\NodeJS\WeatherApp\web-server\src
ucu 3000 portunu dinliyor
```

Title bilgisi değişken hale gelecek ve sayfalarımıza bakalım.



Uygulamamızı daha kullanışlı hale getirmek için sayfalar arasında dolaşmak için link ekleyelim.

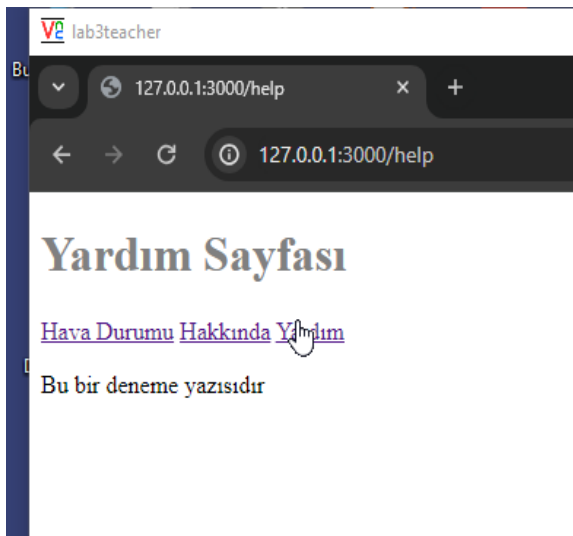
```
JS app.js ...src header.hbs help.hbs index.hbs a
web-server > public > templates > partials > header.hbs > div > a
1 <h1>{{title}}</h1>
2
3 <div>
4   <a href="/">Hava Durumu</a>
5
6 </div>
```

Link eklemek için a etiketi kullanılır toplamda 3 tane link yazalım. İlk a etiketi ana sayfa içindir. A etiketleri arasında linkler üzerinde görölmesini istediğimiz yazıları ekleyelim.

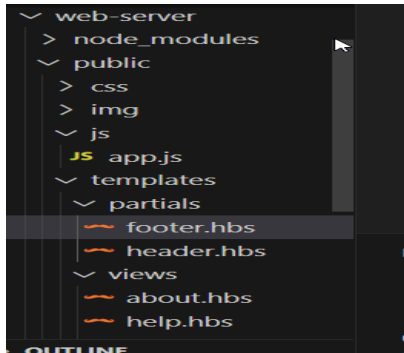
```
JS app.js ...src header.hbs help.hbs index.hbs about.hbs
web-server > public > templates > partials > header.hbs > div > a
1 <h1>{{title}}</h1>
2
3 <div>
4   <a href="/">Hava Durumu</a>
5   <a href="/about">Hakkında</a>
6   <a href="/help">Yardım</a>
7 </div>
```

Kaydedelim ve tarayıcıdan sayfalarımıza bakalım. Linkler üzerine tıklayalım yönlendirme işlemi yapacaktır.





Linklere tıkladığımızda sayfalar arasında dolaşmamıza imkân tanır. Web sayfamızda tekrar eden kısımları daha kullanışlı hale getirir. Tekrar eden kısımlar partials klasörüne dahil edilir. Patials içinde yeni bir dosya açalım footer.hbs isminde.



İçine şu kodu yazalım ve kaydedelim.

```
JS app.js ...src header.hbs footer.hbs help.hbs
web-server > public > templates > partials > footer.hbs > p
1 <p>{{name}} tarafından geliştirilmiştir</p>
abc geliştirilmiştir
```

Teker teker dosyalara ekleyelim. Help.hbs içine gidelim body etiketi kapanmadan önce footer i ekleylim.

```
JS app.js ...src header.hbs footer.hbs help.hbs index.hbs
web-server > public > templates > views > help.hbs > html > body
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    <p>{{helpText}}</p>
11    {{>footer}}
12  </body>
13 </html>
```

About.hbs nin eski hali sil bunu ve footer ekle diğer sayfadaki gibi.

```
header.hbs footer.hbs help.hbs index.hbs about.hbs X
web-server > public > templates > views > about.hbs > html > body > p
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    
11    <p>{{name}} tarafından geliştirilmiştir</p>
12  </body>
```

```
server > public > templates > views > index.hbs > html > body

<html>
  <head>
    <link rel="stylesheet" href="/css/styles.css">
    <script src="/js/app.js"></script>
  </head>

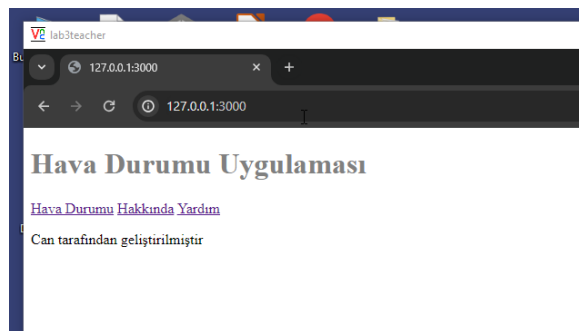
  <body>
    {{>header}}
    {{>footer}}
  </body>
</html>
```

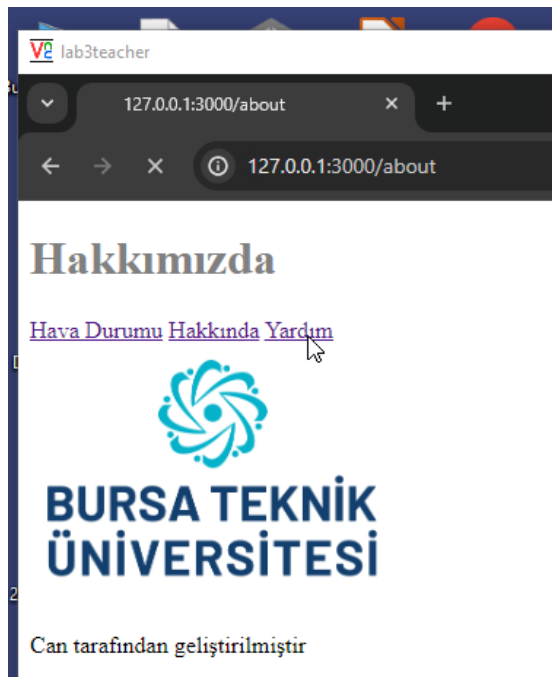
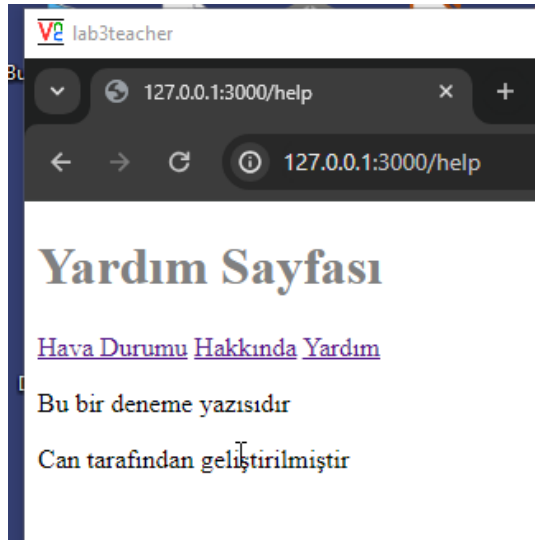
Name bilgisinin app.js içinde her route için gönderilmiş olması gerekli. İndex ve about sayfalarında name bulunuyor fakat help de bulunmuyordu buraya name kısmını ekleyelim.

```
JS app.js ...src X header.hbs footer.hbs help.hbs index.hbs
web-server > src > JS app.js > app.get('/help') callback > name

33  })
34
35  app.get('/help', (req, res) => {
36    res.render('help', {
37      title: 'Yardım Sayfası',
38      name: 'Can',
39      helpText: 'Bu bir deneme yazısıdır'
40    })
41  })
42  // app.com: app.com, app.com/help, app.com/about
43
```

Tarayıcıdan sayfalarımızı kontrol edelim.





Böylece sayfalarımız arasında tutarlılık oluşturmuş olduk yeni bir sayfa eklediğimizde yapmamız gereken tek şey içeriğinin öncesine header sonrasında footer eklememiz gereklidir. Bizim app.js içinde toplamda üç adet router bilgimiz bulunmaktadır.

```
JS app.js ...src header.hbs footer.hbs help.hbs index.hbs
web-server > public > templates > views > help.hbs > html > body > p
3 <html>
4 <head>
5   <link rel="stylesheet" href="/css/styles.css">
6 </head>
7
8 <body>
9   {{>header}}
10  <p>{{helpText}}</p>
11  {{>footer}}
12 </body>
13 </html>
```

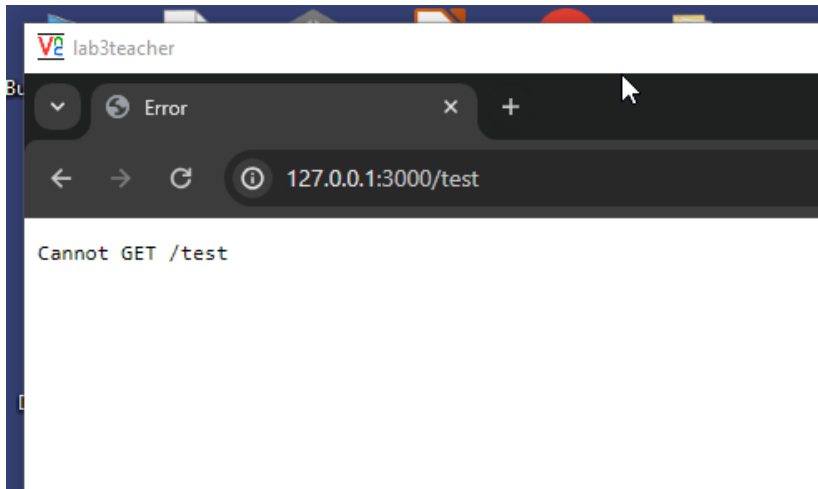

Anasayfamızın router'i:

```
app.get('/', (req, res) => {  
  res.render('index', {  
    title: 'Hava Durumu Uygulaması',  
    name: 'Can'  
  })  
})
```

About sayfamızın router'i:

```
app.get('/about', (req, res) => {  
  res.render('about', {  
    title: 'Hakkımızda',  
    name: 'Can'  
  })  
})
```

Diyelim ki ziyaretçimiz olmayan bir sayfaya erişim yaparsa karşısına şu çıkar:



Default bir hata yazısı çıkar. Bu yazı yerine daha kullanışlı bir uyarı yazısı yapalım. App.js dosyamız şu an bu şekildedir.

```
JS app.js ...src X header.hbs footer.hbs help.hbs
web-server > src > JS app.js > ...
18 const partialsPath = path.join(__dirname, '../public/tem
19 hbs.registerPartials(partialsPath)
20
21 app.get('/', (req, res) => {
22   res.render('index', {
23     title: 'Hava Durumu Uygulaması',
24     name: 'Can'
25   })
26 })
27
28 app.get('/about', (req, res) => {
29   res.render('about', {
```

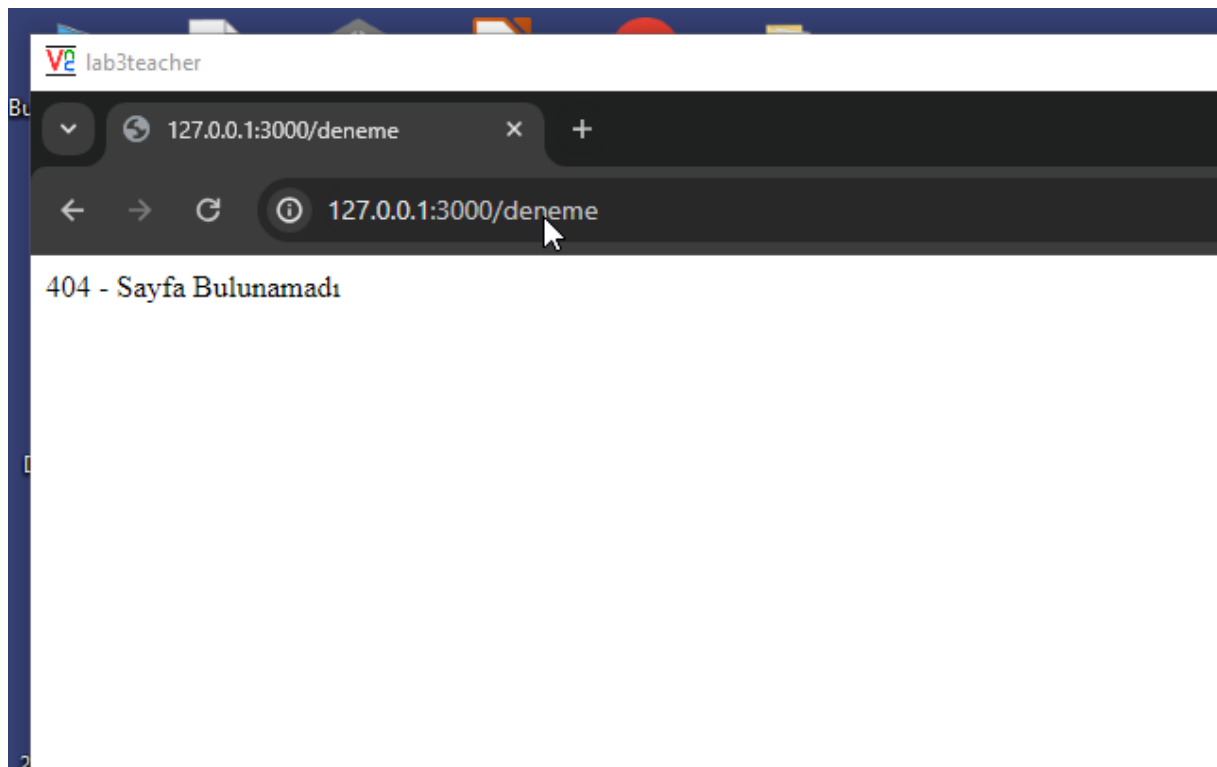
En son sayfadan sonra değişiklik yapılır yeri her zaman en sondur ve app.get() içinde * kullanılır. * ın anlamı ise * yerine herhangi bir şey gelebilir anlamına gelmektedir. *: ne olursa olsun anlamına gelir.

```
JS app.js ...src X header.hbs footer.hbs help.hbs index.hbs
web-server > src > JS app.js > ...
55 // app.get('/help', (req, res) => {
56 //   res.send([name: 'Can'],
57 //     {name: 'Canan'}})
58 // })
59
60 // app.get('/about', (req, res) => {
61 //   res.send('<h1>Hakkımızda sayfası!</h1>')
62 // })
63
64 app.listen(3000, () => {
65   console.log('Sunucu 3000 portunu dinliyor')
66 })
```

```
JS app.js ...src header.hbs footer.hbs help.hbs index.hbs
web-server > src > JS app.js > app.get(*) callback
55 // app.get('/help', (req, res) => {
56 //   res.send([name: 'Can'],
57 //     {name: 'Canan'}})
58 // })
59
60 // app.get('/about', (req, res) => {
61 //   res.send('<h1>Hakkımızda sayfası!</h1>')
62 // })
63
64 app.get('*', (req, res) => {
65   res.send('404 Sayfa Bulunamadı')
66 })
```

Test edelim arama kısmına bulunmayan bir sayfa ismi ile çağrıda bulunalım mesela deneme ile.

Html de eğer bir sayfa bulunamadıysa bu 404 sayfası olarak adlandırılır.



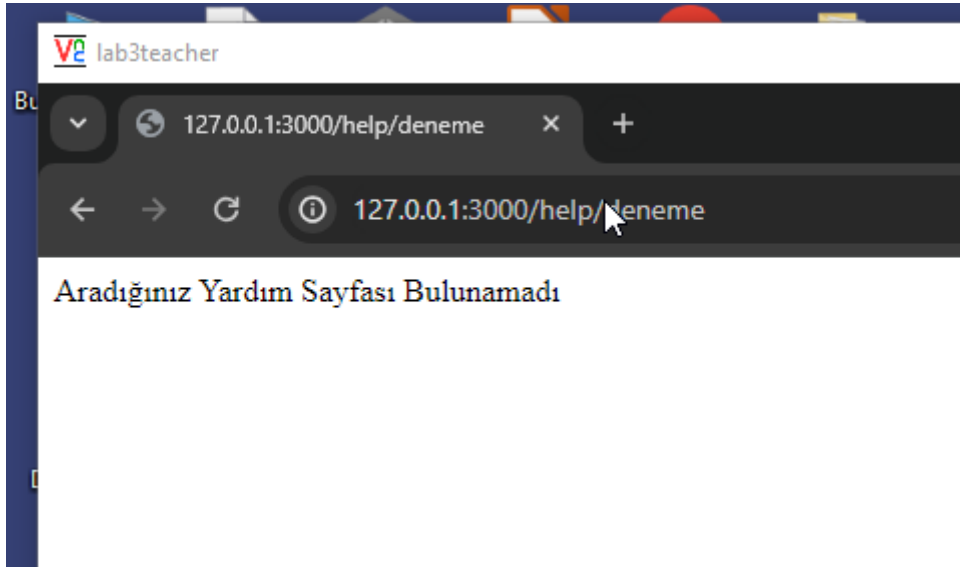
Daha kullanışlı hale getirelim. Elimizde help.hbs dosyamız var eğer help sayfası bulunamadıysa yani helpden sonra bulunmayan bir sayfa giriş yaparsa help/* kullanırız ve aradığınız yardım sayfası bulunamadı yazısını basalım.

```
JS app.js ...src X header.hbs footer.hbs help.hbs in
web-server > src > JS app.js > app.get('/help/*') callback
59
60 // app.get('/about', (req, res) => {
61 //     res.send('<h1>Hakkımızda sayfası!</h1>')
62 // })
63
64 app.get('/help/*', (req, res) => {
65     res.send('Aradığınız Yardım Sayfası Bulunamadı')
66 })
67
68 app.get('*', (req, res) => {
69     res.send('404 - Sayfa Bulunamadı')
70 })
```

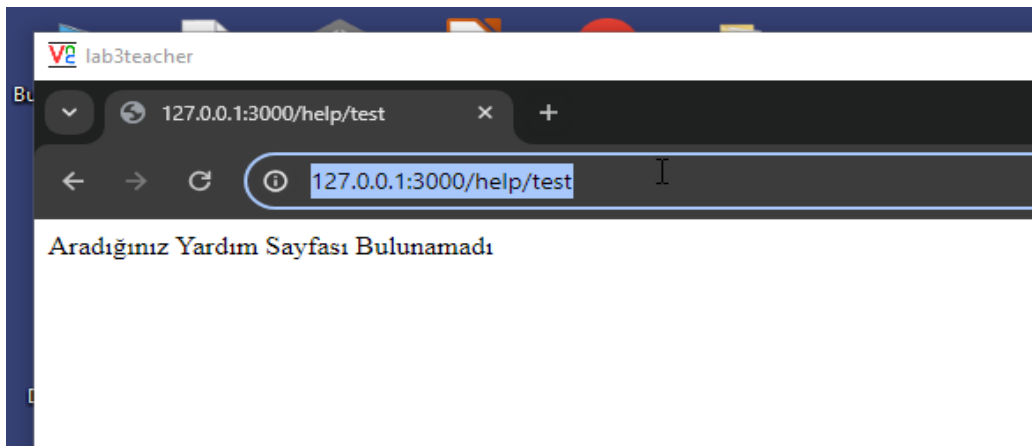
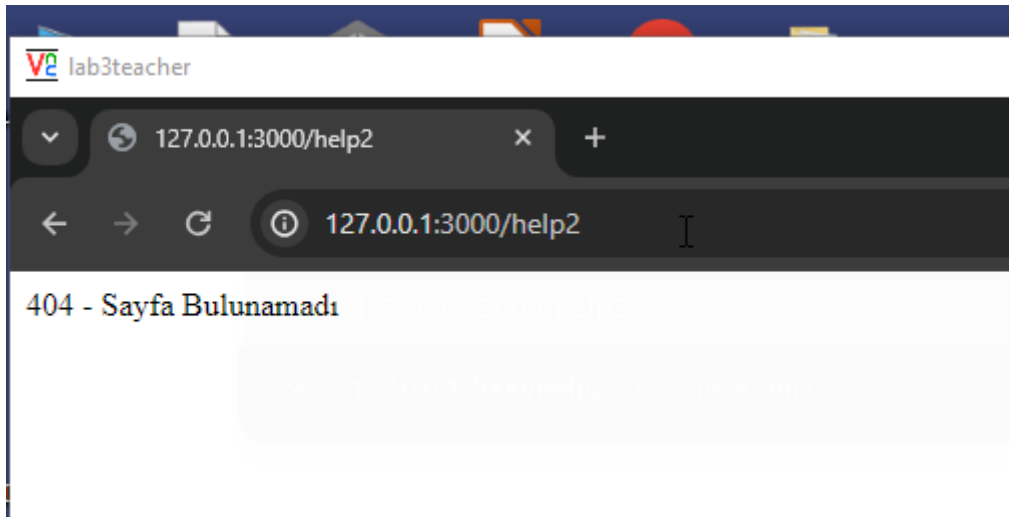
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS node - web-server +

```
[nodemon] restarting due to changes...
[nodemon] starting `node .\src\app.js`
```

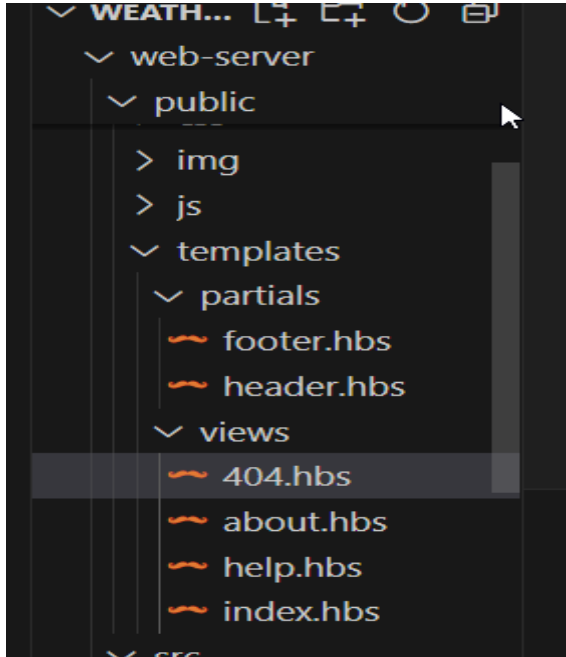
Kodu kaydedelim ve tarayıcıya help/deneme yazalım.



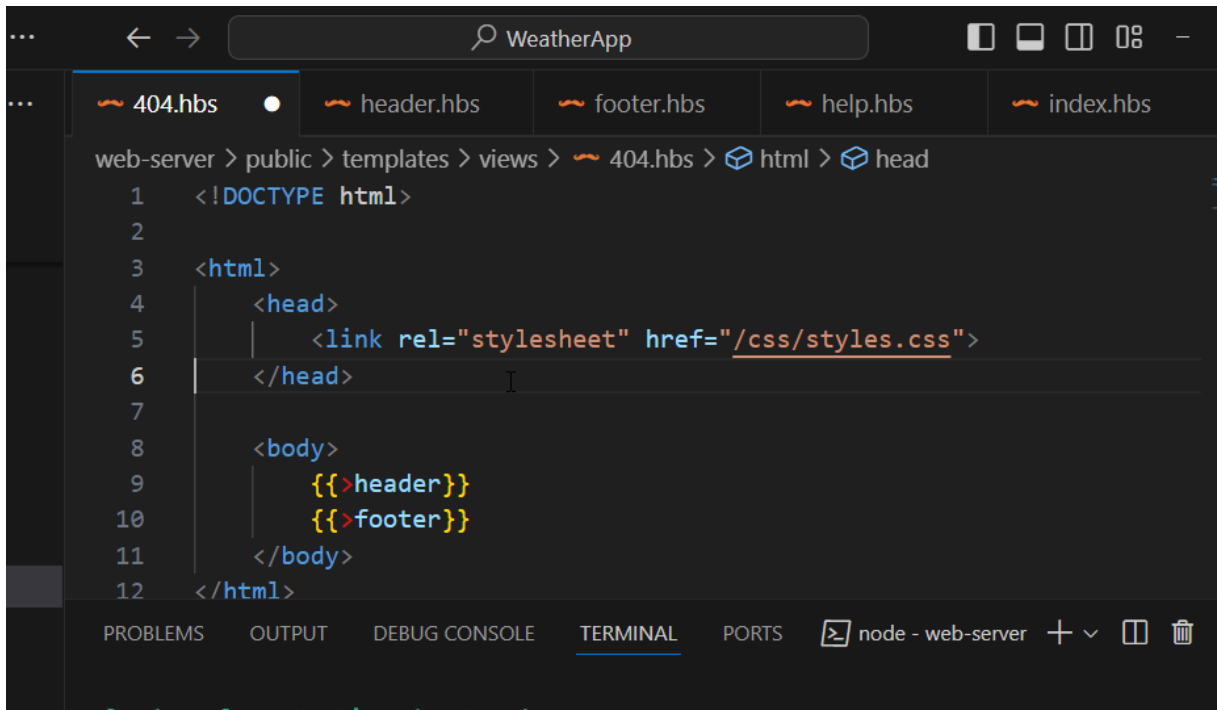
Help2 diye bir sayfamız bulunmadığı için aradığınız yardım sayfası bulunamadı ifadesi değil de 404 sayfa bulunamadı yazısını görürüz.



Yeni bir dosya açalım adı 404.hbs olsun.



404.hbs dosyasının içeriğini oluşturalım.



Biraz düzenleme yapalım.

```
404.hbs header.hbs footer.hbs help.hbs index.hbs
web-server > public > templates > views > 404.hbs > html > body > p
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    <p>{{errorMessage}}</p>
11    {{>footer}}
12  </body>
13 </html>
```

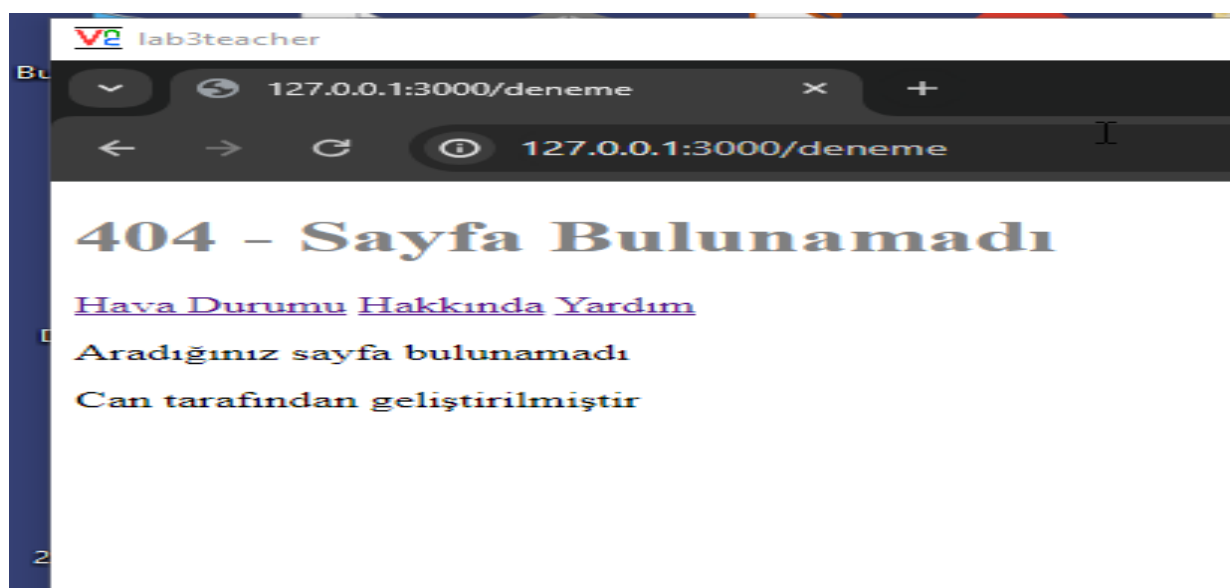
App.js içinde bazı değişiklikler yaparak 404 dosyasındaki error mesajı nasıl ekrana basabiliriz bunu deneyelim. App.js dosyasının eski hali:

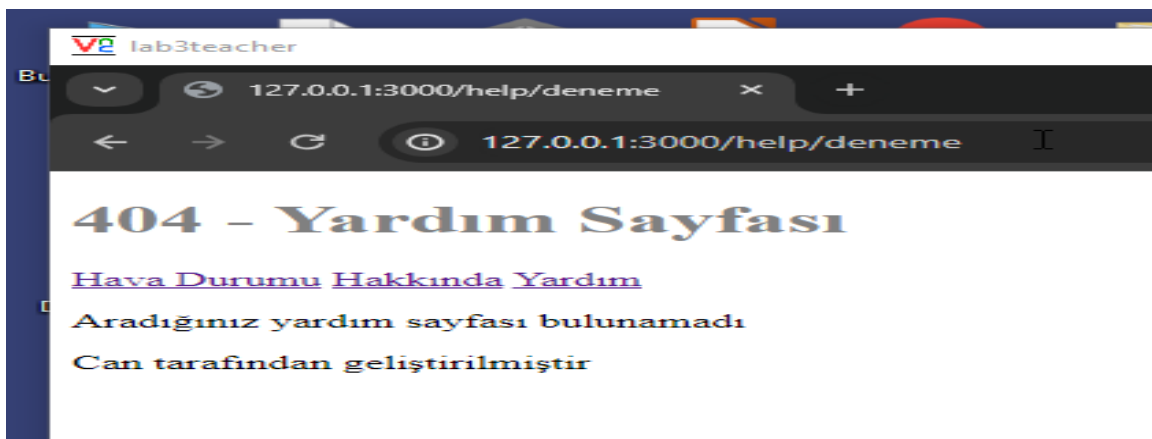
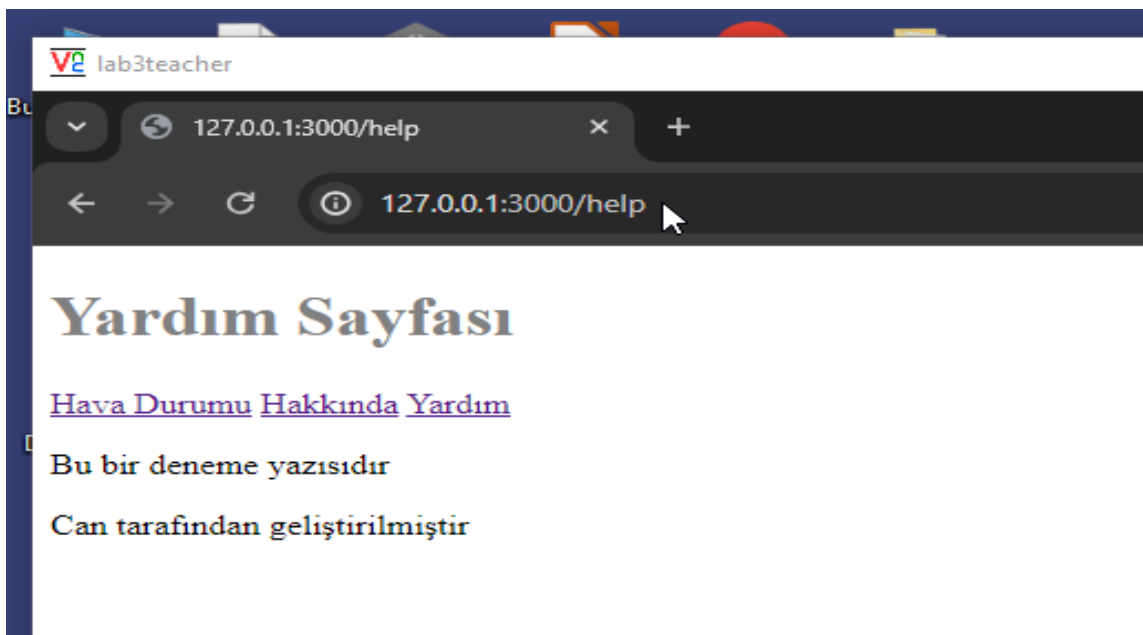
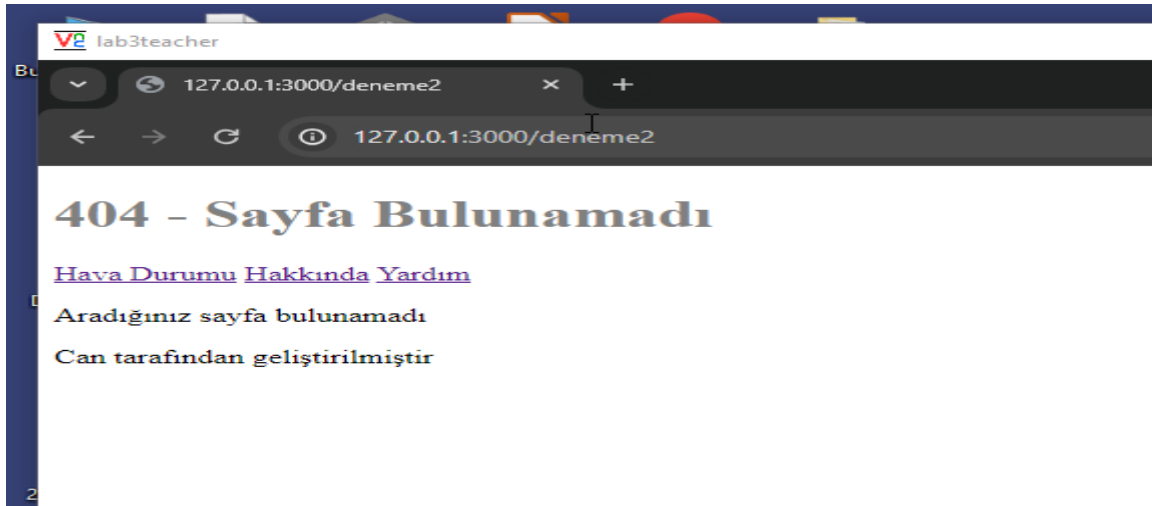
```
JS app.js ...\src 404.hbs header.hbs footer.hbs help.hbs
web-server > src > JS app.js > ...
43 app.get('/weather', (req, res) => {
44   forecast: 'Hava yağışlı',
45   location: 'Bursa'
46 })
47 })
48 })
49
50 app.get('/help/*', (req, res) => {
51   res.send('Aradığınız Yardım Sayfası Bulunamadı')
52 })
53
54 app.get('*', (req, res) => {
55   res.send('404 - Sayfa Bulunamadı')
56 })
```

Yapmamız gereken ilk değişiklik `res.send()` değil de `res.render()` ifadesi kullanmamız gerekli çünkü hbs sayfaları render edilir. Help ve * kısımlarında bunu yapalım ardından hbs sayfalarının bizden beklediği bilgiler var mesela title, name bilgisi gibi. `Res.render()` kısmına ilk parametre olarak kullanmak istediğimiz hbs dosyamızın adıdır.

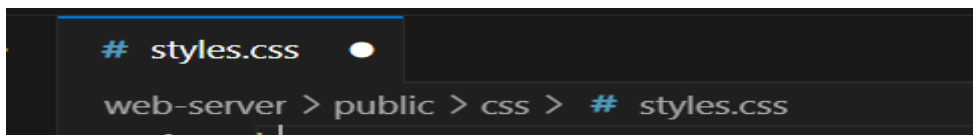
```
web-server > src > JS app.js > app.get('*') callback
45 app.get('/weather', (req, res) => {
46
47 })
48
49
50 app.get('/help/*', (req, res) => {
51   res.render('404', {
52     title: '404 - Yardım Sayfası',
53     name: 'Can',
54     errorMessage: 'Aradığınız yardım sayfası bulunamadı'
55   })
56 })
57
58 app.get('*', (req, res) => {
59   res.render('404', {
60     title: '404 - Sayfa Bulunamadı',
61     name: 'Can',
62     errorMessage: 'Aradığınız sayfa bulunamadı'
63   })
64 })
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Kaydedelim ve tarayıcıda deneyelim.

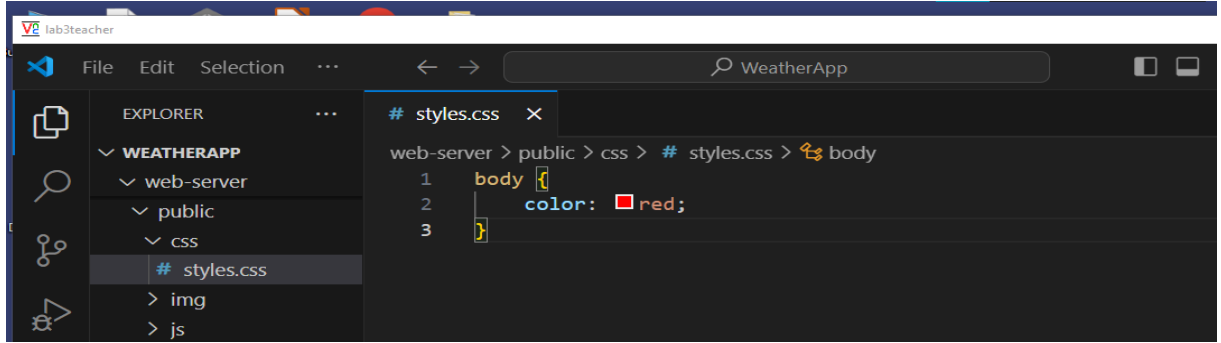




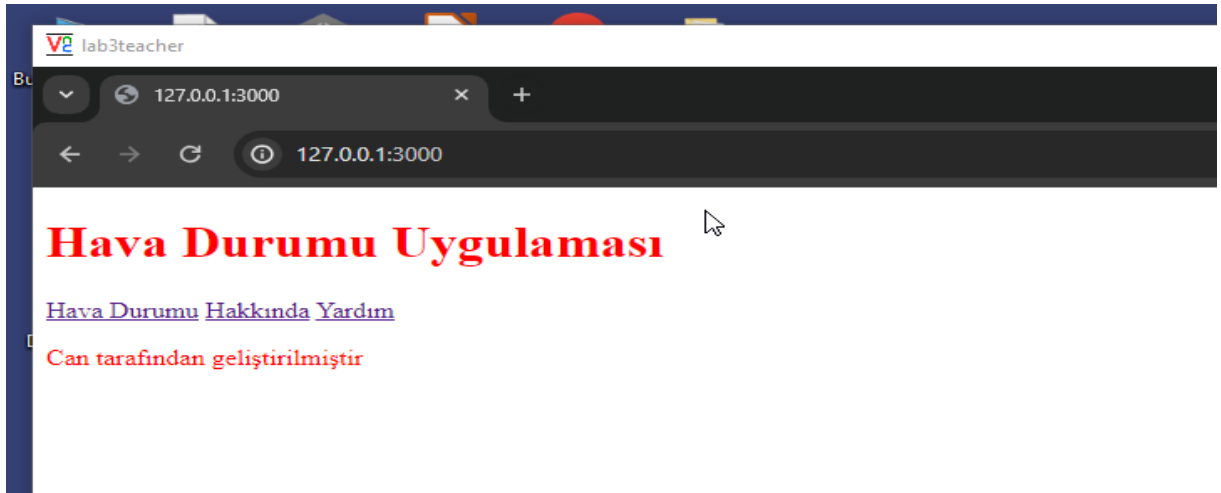
Css dosyası ile işlem yapmaya devam edelim. Daha profesyonel hale getirelim.



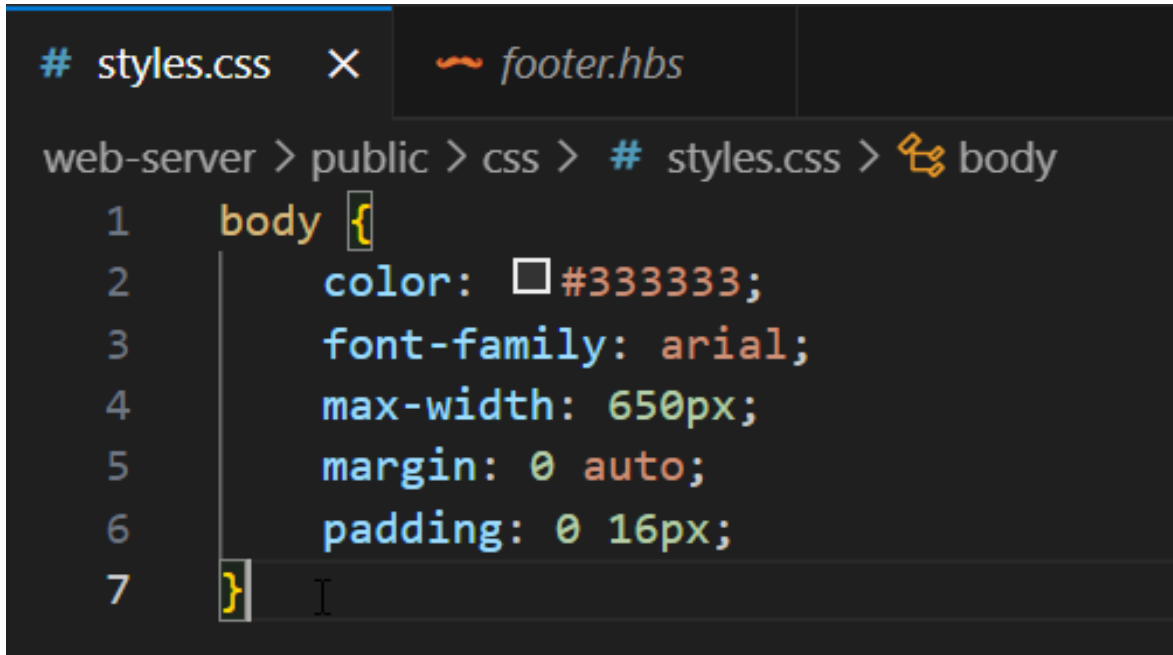
İlk olarak styles.css dosyasının içeriğini tamamen silelim ve sadece renk bilgisini yazalım. Kaydedelim ve tarayıcıdan sayfamıza bakalım.



```
# styles.css
web-server > public > css > # styles.css > body
1 body {
2   color: red;
3 }
```

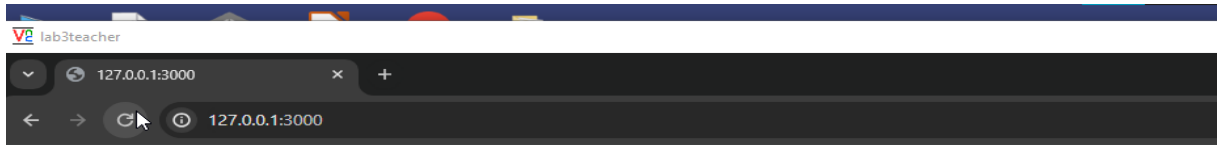


Kırmızı yerine daha iyi görünüme sahip bir renk kullanalım hexadecimal olarak renk kodunu verelim.



```
# styles.css
web-server > public > css > # styles.css > body
1 body {
2   color: #333333;
3   font-family: arial;
4   max-width: 650px;
5   margin: 0 auto;
6   padding: 0 16px;
7 }
```

Margin 0 yukarıdan aşağıya auto ise sağdan soldan ayarlama yapar. Sayfamıza bakalım.

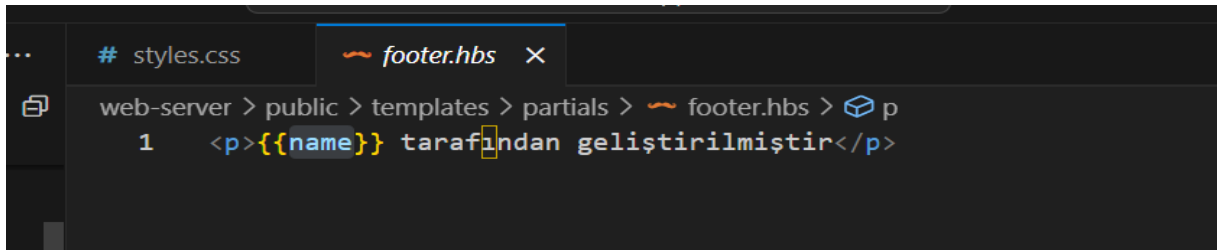


Hava Durumu Uygulaması

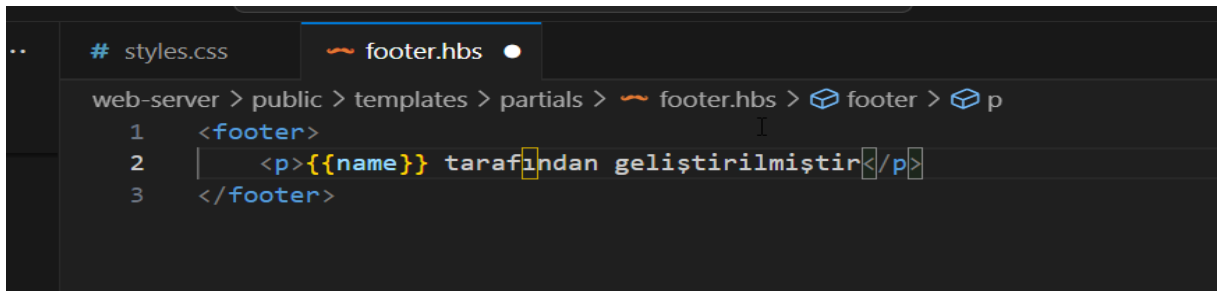
[Hava Durumu Hakkında Yardım](#)

Can tarafından geliştirilmiştir

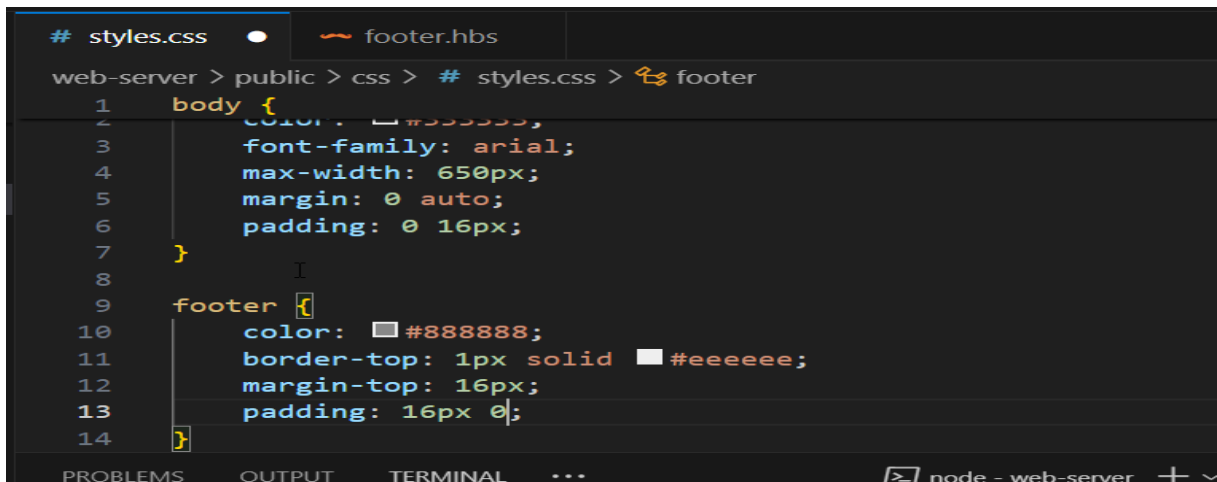
Footer.hbs içinde sadece p etiketi vardır. style.css de sayfanın hangi kısmı ile alakalı konfigürasyon yaptığımızı belirtiriz biz body için özellikler verdik.



Biz sayfanın footer kısmında değişiklik yapmak istiyoruz ama footer kısmının olduğunu nasıl belirtebiliriz? Burada değişiklik yapalım. P etiketlerini footer etiketi içinde oluşturmamız gereklidir.

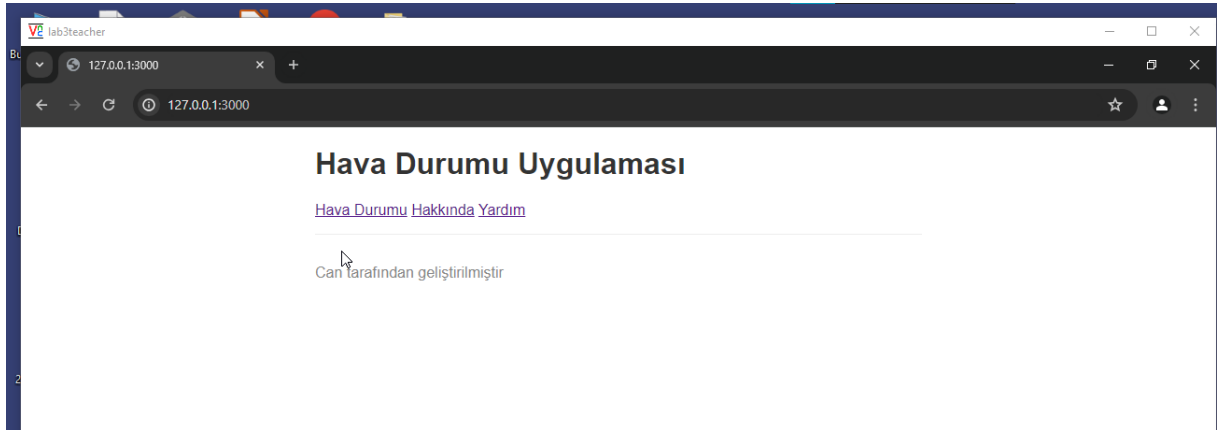


Style.css dosyasına gelelim zaten daha önce body kısmını tanımlamıştık. Footer özelliğini tanımlayalım.

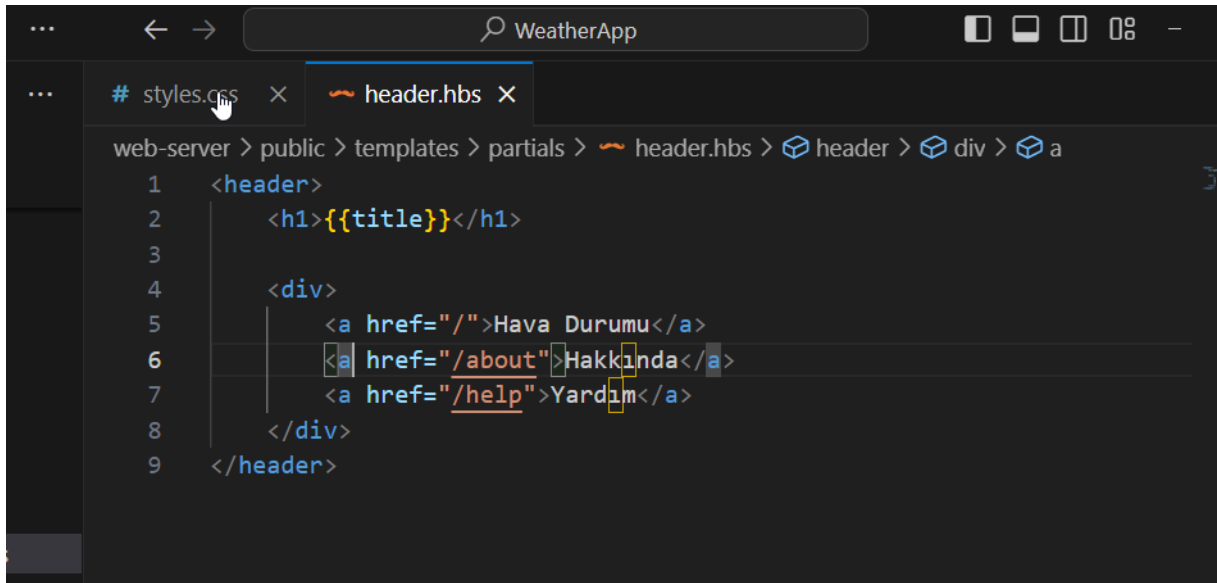


Solid: düz çizgi

Tarayıcıdaki görüntüsüne bakalım.

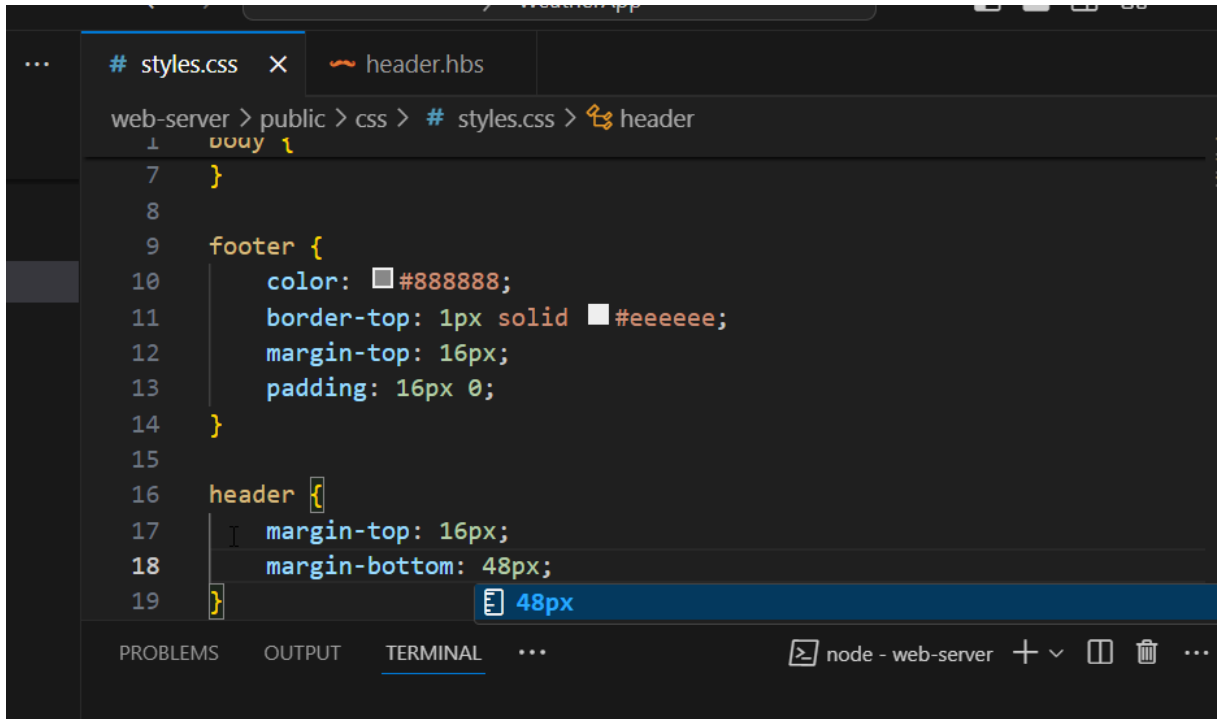


Yaptığımız işlemleri header.hbs içinde yapalım.



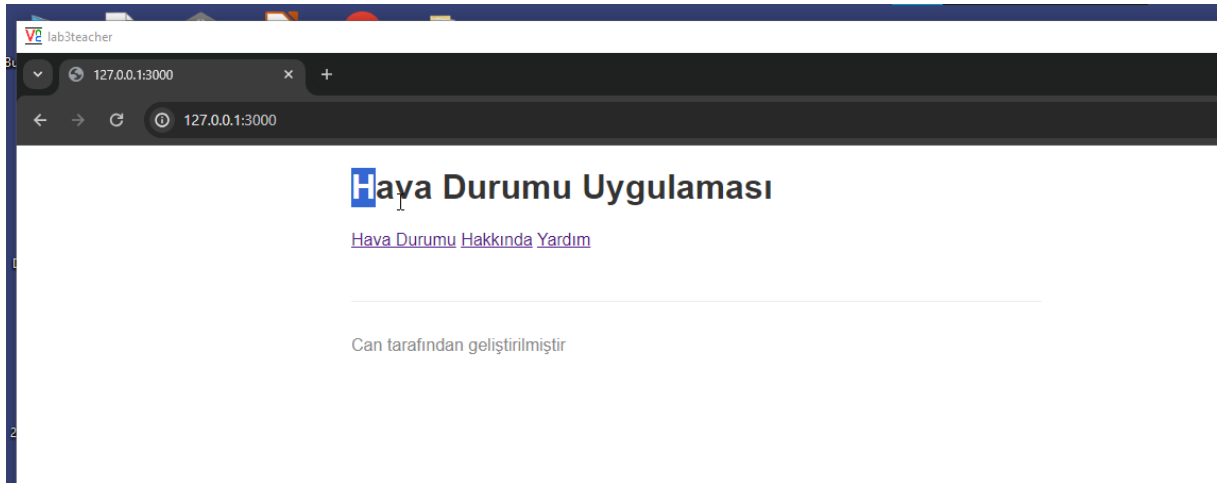
{{header}} etiketini ekleyelim.

Style.css dosyasında etiket özelliklerini verelim.

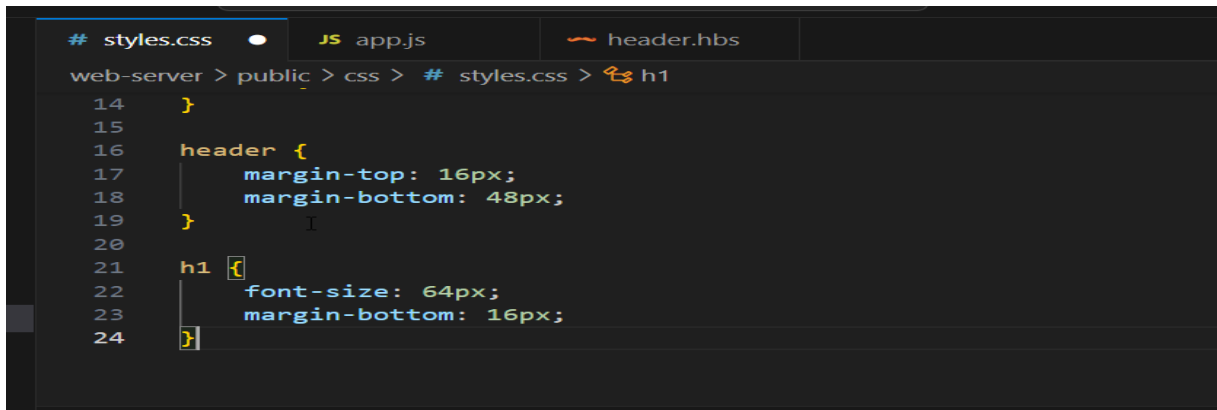


```
# styles.css
web-server > public > css > # styles.css > header
1  body {
7  }
8
9  footer {
10   color: #888888;
11   border-top: 1px solid #eeeeee;
12   margin-top: 16px;
13   padding: 16px 0;
14 }
15
16 header {
17   margin-top: 16px;
18   margin-bottom: 48px;
19 }
```

Tarayıcıdaki görüntüsüne bakalım.



Bir sonraki yapacağımız işlem ise h1 etiketi için özellik ekleyelim.



```
# styles.css
web-server > public > css > # styles.css > h1
14 }
15
16 header {
17   margin-top: 16px;
18   margin-bottom: 48px;
19 }
20
21 h1 {
22   font-size: 64px;
23   margin-bottom: 16px;
24 }
```

Linklerimize özellik ekleyelim. Header a demek header içindeki a için özellik ayarlaması yapar.

```
.. # styles.css JS app.js header.hbs
web-server > public > css > # styles.css > header a
16 header {
17   margin-bottom: 40px;
18 }
19
20
21 h1 {
22   font-size: 64px;
23   margin-bottom: 16px;
24 }
25
26 header a {
27   color: #888888;
28   margin-right: 16px;
29   text-decoration: none;
30 }
```



Linklerin rengi değişti altında çizgi (decoration ile) yoktur. Link olduğunu anlamamız için üstüne gitmemiz lazım. Resim kullandığımız about.hbs sayfamıza gidelim. Sadece about sayfamızdaki resme özellik vermek istersek şu adımları takip etmeliyiz.

```
about.hbs JS app.js header.hbs
public > templates > views > about.hbs > html > body > img.portrait
html>
<head>
  <link rel="stylesheet" href="/css/styles.css">
</head>

<body>
  {{>header}}
  
  {{>footer}}
</body>
html>
```

İmg etiketine class bilgisi verilir. Şu anda resmimiz bu şekilde görünüyor.



Şimdi style.css dosyası içinde .portrait ile özellik ekleyelim.

```
web-server > |
21  h1 { (Edge 12, Firefox 1, Safari 1, Chrome 1, IE 3, Opera 3)
24  }
25
26  head MDN Reference
27  color: #888888;
28  margin-right: 16px;
29  text-decoration: none;
30  }
31
32  .portrait {
33  width: 250px;
34  }
```

PROBLEMS OUTPUT TERMINAL ...

node - web-s

C:\Users\hilmut\Documents\NodeJS\WeatherApp\web-server\src

Kodumuzu kaydedelim ve tarayıcıda çalıştıralım.



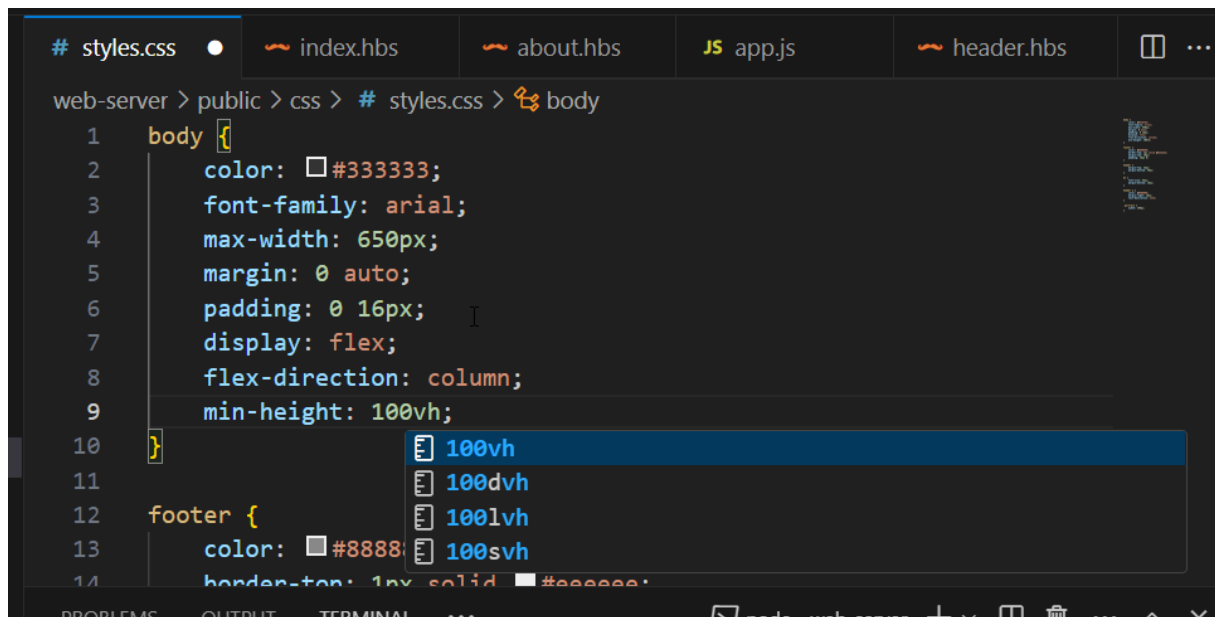
Değişik sayfalar arasına gittiğimizde footer yukarı doğru kayar bunu sayfanın altına sabitlememiz lazım. Index.hbs ye gidelim body içinde div oluşturup bu diz için class tanımlayalım. Index.hbs nin ilk hali:

```
Selection  ...  WeatherApp
# styles.css  index.hbs  about.hbs  JS app.js  header.hbs
web-server > public > templates > views > index.hbs > html
3  <html>
4    <head>
5      <link rel="stylesheet" href="/css/styles.css">
6      <script src="/js/app.js"></script>
7    </head>
8
9    <body>
10     {{>header}}
11     {{>footer}}
12   </body>
13 </html>
```

Kod eklenmiş hali:

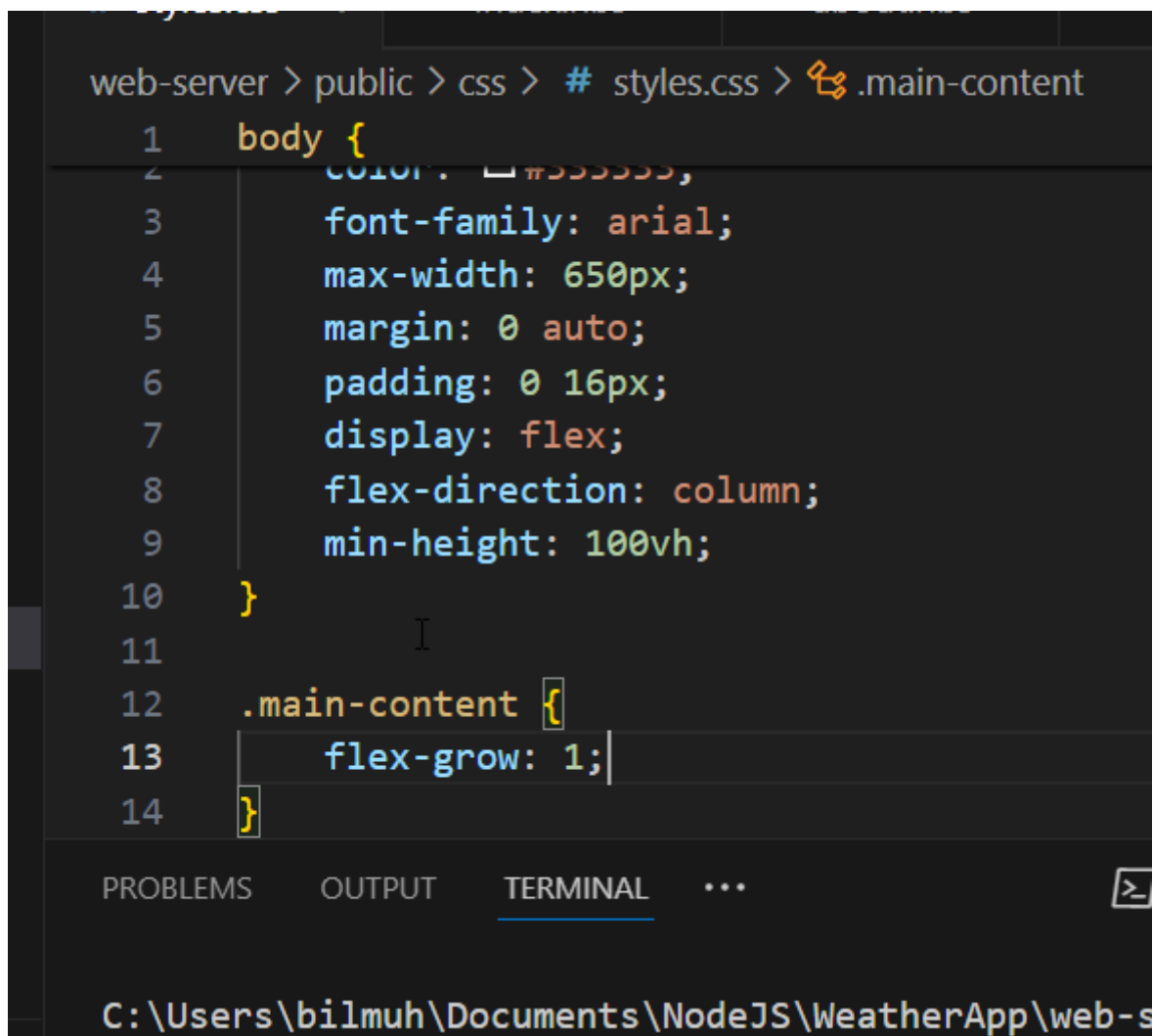
```
# styles.css  index.hbs  about.hbs  JS app.js  header.hbs
web-server > public > templates > views > index.hbs > html > body > div.main-content > p
3  <html>
4    <head>
5      <link rel="stylesheet" href="/css/styles.css">
6      <script src="/js/app.js"></script>
7    </head>
8
9    <body>
10     <div class="main-content">
11       {{>header}}
12       <p>Hava durumu bilgisini öğrenmek için web sitemizi kullanın</p>
13     </div>
14
15     {{>footer}}
```

P etiketi ile bilgilendirme mesajı ekleyelim. Böylece footeri diğer özelliklerden ayırmış olduk. Daha sonra css dosyasında body içinde düzenleme yapalım. Display: flex yazalım. Bunun ile box oluşturmuş olur.



```
# styles.css
web-server > public > css > # styles.css > body
1 body {
2   color: #333333;
3   font-family: arial;
4   max-width: 650px;
5   margin: 0 auto;
6   padding: 0 16px;
7   display: flex;
8   flex-direction: column;
9   min-height: 100vh;
10 }
11
12 footer {
13   color: #888888;
14   border-top: 1px solid #000000;
```

100vh tamamını kullan demektir. Main-content içinde ekleme yapalım. Tasarımda classlar . ile tanımlanır.

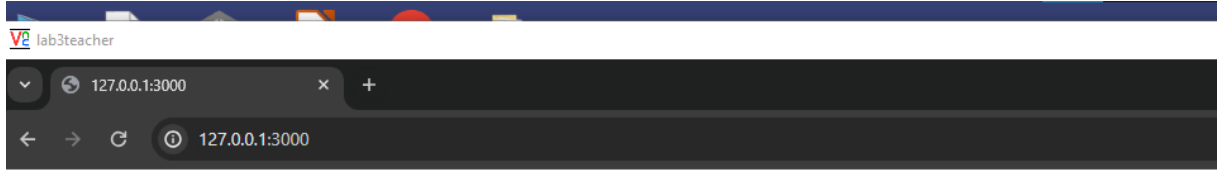


```
web-server > public > css > # styles.css > .main-content
1 body {
2   color: #333333;
3   font-family: arial;
4   max-width: 650px;
5   margin: 0 auto;
6   padding: 0 16px;
7   display: flex;
8   flex-direction: column;
9   min-height: 100vh;
10 }
11
12 .main-content {
13   flex-grow: 1;
14 }
```

PROBLEMS OUTPUT TERMINAL ...

C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-s

Kaydedelim ve tarayıcı da sayfalarımıza bakalım sonuçta sayfalar arasında gezerken sayfaların doluluklarına bağlı olarak sayfanın footerı artık yukarı kaymamaktadır.

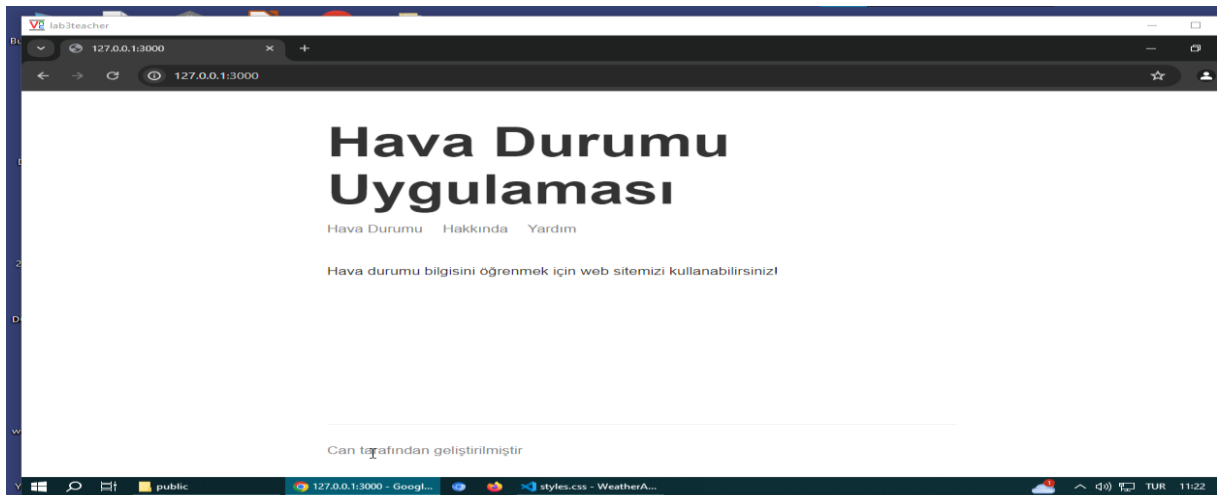


Hava Durumu Uygulaması

[Hava Durumu](#) [Hakkında](#) [Yardım](#)

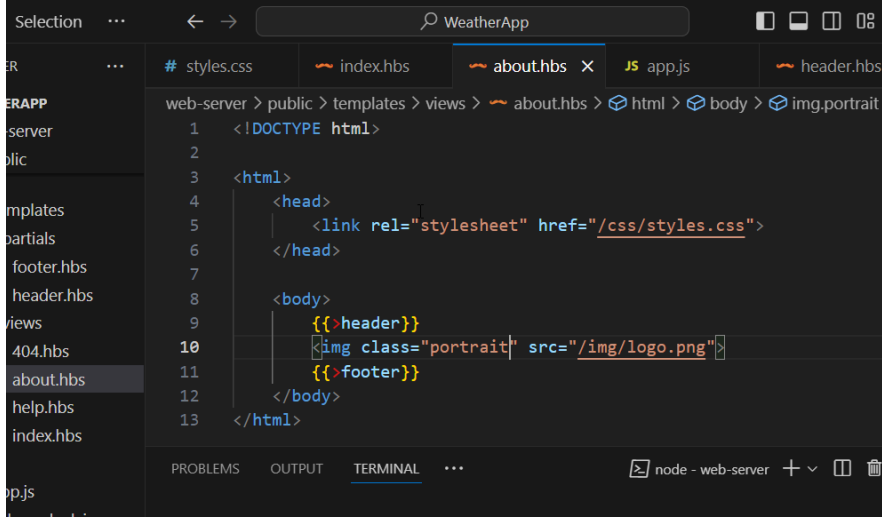
Hava durumu bilgisini öğrenmek için web sitemizi kullanabilirsiniz!

About ve help dosyalarımız da bu değişiklikleri yapmadığımız için footer yukarı kaymaktadır.



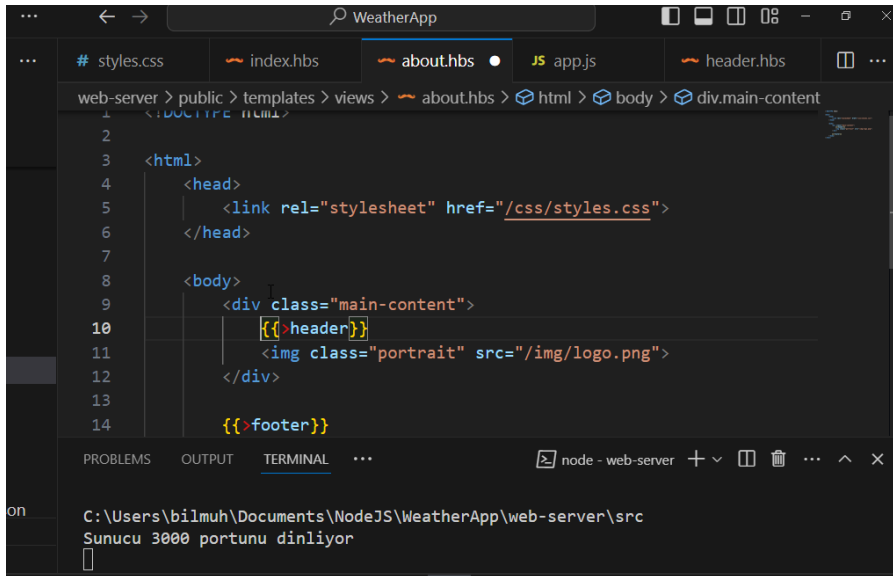
Index.hbs içinde yaptığımız gibi about.hbs için bir div tanımlaması ve içinde class tanımlayalım.

About.hbs nin ilk hali:



```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     {{>header}}
10    
11    {{>footer}}
12  </body>
13 </html>
```

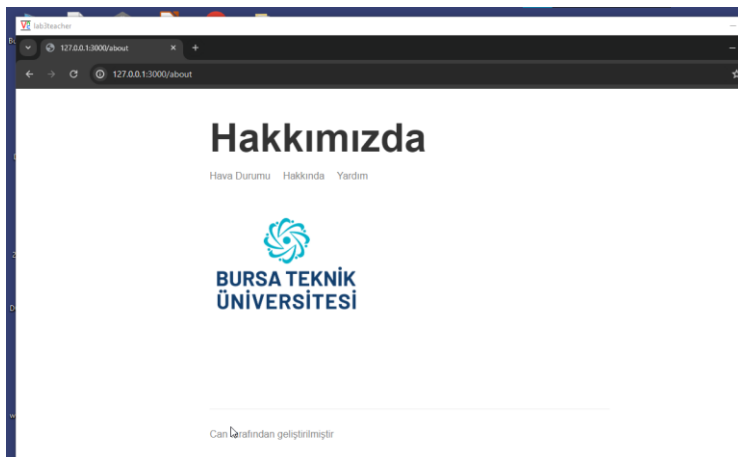
Div eklenmiş hali:



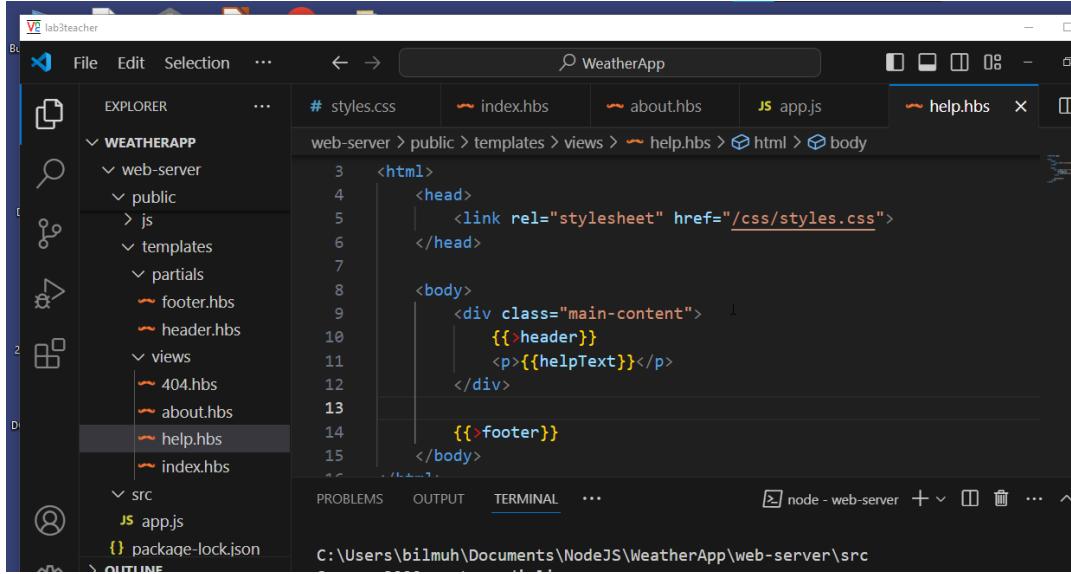
```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     <div class="main-content">
10      {{>header}}
11      
12    </div>
13
14    {{>footer}}
```

C:\Users\bilmuh\Documents\NodeJS\WeatherApp\web-server\src
Sunucu 3000 portunu dinliyor

Kodu kaydedelim ve tarayıcıda bakalım.



Yaptığımız bu işlemleri help.hbs içinde tekrar edelim.

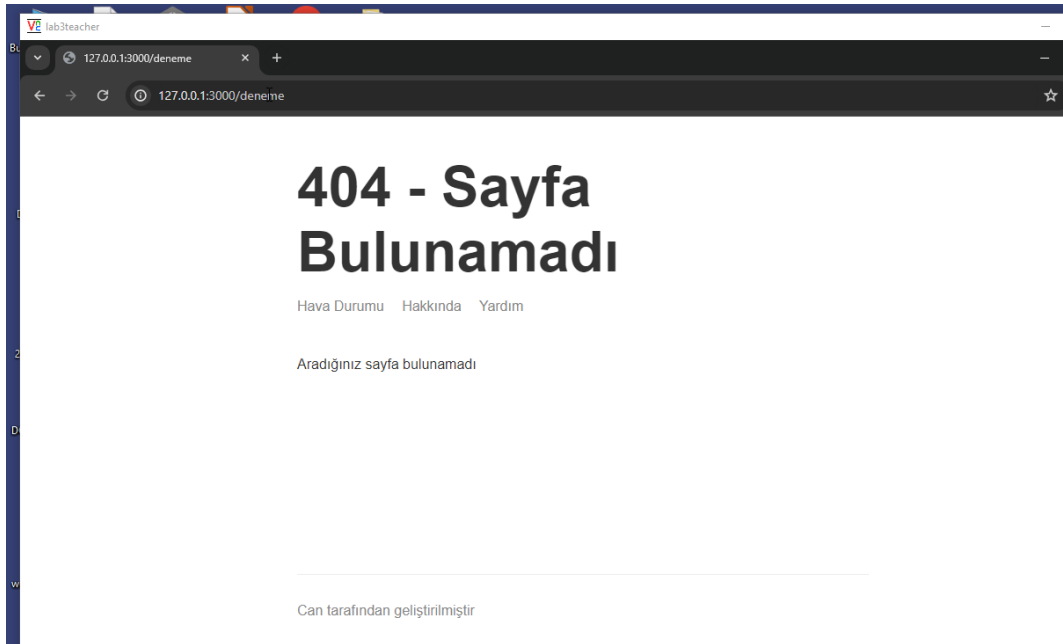


The screenshot shows the Visual Studio Code editor with the 'WeatherApp' project open. The Explorer sidebar on the left shows the file structure: 'web-server' > 'public' > 'templates' > 'views' > 'help.hbs'. The main editor area displays the content of 'help.hbs' with the following code:

```
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6   </head>
7
8   <body>
9     <div class="main-content">
10      <div>
11        <p>{{header}}</p>
12      </div>
13
14      <div>
15        <p>{{helpText}}</p>
16      </div>
17    </div>
18    <div>
19      <p>{{footer}}</p>
20    </div>
21  </body>
22</html>
```

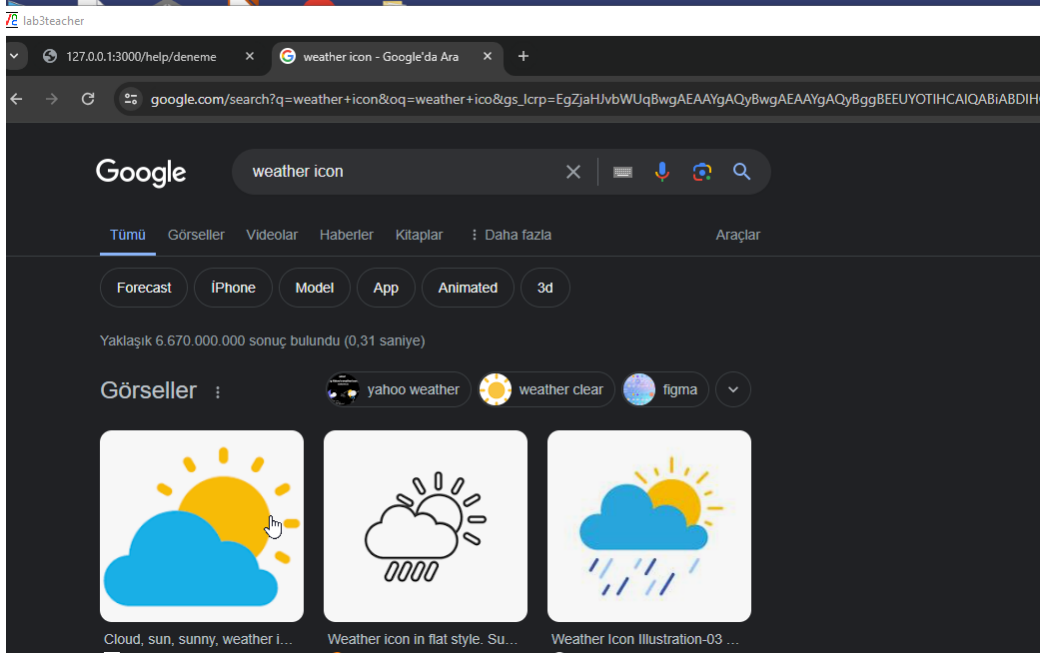
The bottom status bar shows the file path: 'C:\Users\bilmut\Documents\NodeJS\WeatherApp\web-server\src'.

Bu işlemleri 404 içinde yapalım. Tarayıcıda bulunamayan sayfalara gidelim.

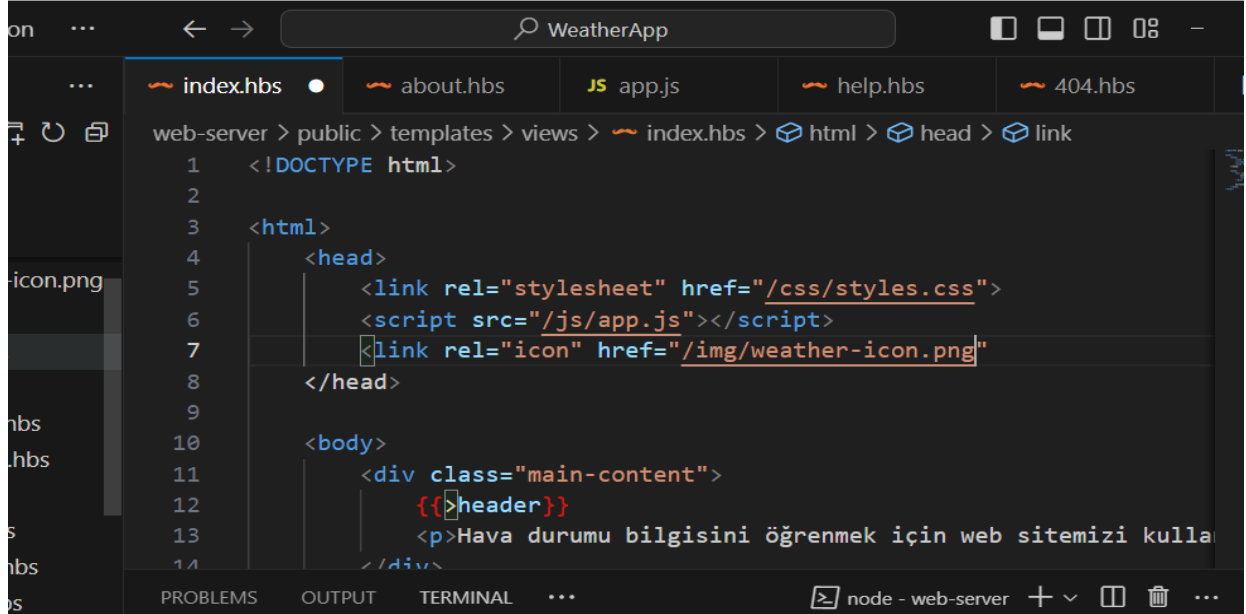




Daha sonra hava durumu ile ilgili icon bulalım.

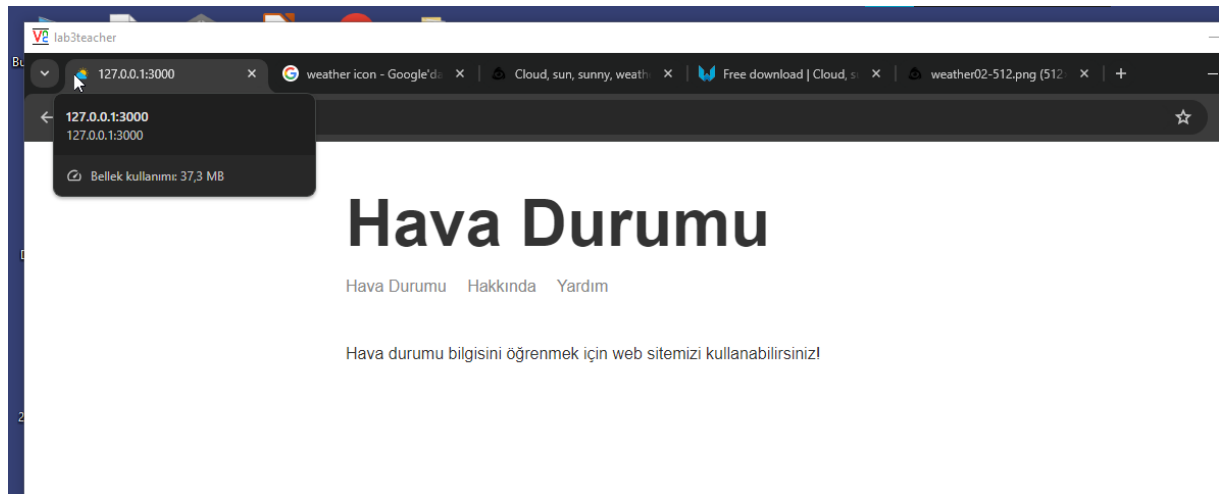


Görseli kaydedelim. İmg klasörü içine indirdiğimiz görseli kaydedelim. index.hbs içine gidelim ve link etiketi ile resmimizi ekleyelim. Link etiketinde rel="icon" ifadesi ile bunun bir icon olduğunu belirtmiş olduk.



```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <link rel="stylesheet" href="/css/styles.css">
6     <script src="/js/app.js"></script>
7     <link rel="icon" href="/img/weather-icon.png"
8   </head>
9
10  <body>
11    <div class="main-content">
12      {{header}}
13      <p>Hava durumu bilgisini öğrenmek için web sitemizi kulla
14    </div>
```

Kaydedelim ve tarayıcıdan bakalım.



Sol en üst köşede bir görselin icon olarak konumlandırıldığını görmüş olduk. Diğer about ve help sayfalarına da bu kodu kopyalayalım.

```
index.hbs  about.hbs X JS app.js  help.hbs  404.hbs
web-server > public > templates > views > about.hbs > html > head > link
1  <!DOCTYPE html>
2
3  <html>
4    <head>
5      <link rel="stylesheet" href="/css/styles.css">
6      <link rel="icon" href="/img/weather-icon.png">
7    </head>
8
9    <body>
10     <div class="main-content">
11       <{{>header}}>
12       
13     </div>
14
```

