Web Programlama
13. Hafta

16.12.2024

Mühendislik Fakültesi
Bilgisayar Mühendisliği

Hazırlayan: Dr. Ercan Ezin

GİRİŞ

Arka Uç Teknolojisi Devam- Oturumlar ve Çerez

Not: Bu dersin içeriği kaynakçada belirtilen materyallerde derlenerek üretilmiştir.
Credit: Content of the course has been picked from various sources which has been generously shared by the authors. See credit page for full list.

NODE.JS: EXPRESS

Node.JS üzerinde web uygulamalarının kolaylıkla yazılması ve yönetilmesi için hazır çerçeveler vardır. Bunlardan en popüleri Express kütüphanesidir. Express kütüphanesi ile Web uygulamalarının ihtiyaç duyduğu güvenlik, API ve performans gibi konularda açık kaynaklı olarak sunulmuş hazır metodlar kullanılır.

Kurulum:

```
$ npm install express
```

NODE.JS: EXPRESS SIMPLE APP

Express kütüphanesini diğer normal kütüphaneler gibi uygulamaya ekleyebilirsiniz. Sonrasında 3000 portu gibi bir portunu dinleyen basit bir uygulama geliştirmek için aşağıdaki gibi bir kod yazıp `$node server.js` komutu ile çalıştırabiliriz.

server.js

```
const express = require('express’)
const app = express()
const port = 3000
app.get('/', (req, res) => {
        res.send('Merhaba Dünya!’)
})

app.listen(port, () => {
        console.log(`Example app listening on port ${port}`)
})
```

EXPRESS: GENERATOR

Express kütüphanesi ile uygulama geliştirirken ihtiyaçlarınıza göre hızlıca bir şablon uygulama oluşturabilirsiniz. Bu sayede ön kurulum gerektiren bazı projeleri yaparken hızlanmış olursunuz.

```
$ npx express-generator  veya $ npm install –g express generator
```

```
NOT: -g parametresi global node.js modüllerine Expressi kurmaya
yarar. Böylelikle gelecekte Express-generator kullanmak
istendiğinde komut satırına direkt express komutu yazarak
işlemler yapabiliriz.
```

EXPRESS GENERATOR OPTIONS

$ express -h

Usage: express [options] [dir] // Kullanım: express [seçenekler] [dizin]

Options: // Seçenekler:

```
  -h, --help          output usage information // Kullanım bilgisini yazdır
      --version       output the version number // Versiyon numarasını yazdır
  -e, --ejs           add ejs engine support // EJS motor desteği ekle
      --hbs           add handlebars engine support // Handlebars motor desteği ekle
      --pug           add pug engine support // Pug motor desteği ekle
  -H, --hogan         add hogan.js engine support // Hogan.js motor desteği ekle
      --no-view       generate without view engine // Görüntü motoru olmadan oluştur
  -v, --view <engine> add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade) // Belirtilen
görüntü motoru desteğini ekle (varsayılan: jade)
  -c, --css <engine>  add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css) //
Belirtilen CSS motoru desteğini ekle (varsayılan: düz CSS)
      --git           add .gitignore // .gitignore dosyasını ekle
  -f, --force         force on non-empty directory // Dolu bir dizine zorla yaz
```

EXPRESS APP INSTALLATION

$ npm install

Ekspress şablon yaratıldıktan sonra yukarıdaki komutu çalıştırmak gerekebilir. Bu sayede Express'in çalışması için bağımlı kütüphaneler indirilir.

Gerekli indirmeler tamamlanıp, güvenlik önerileri giderildikten sonra aşağıdaki komutlarla uygulama çalıştırılır.

MacOS veya Linux için uygulamayı bu komut ile çalıştırın:

$ DEBUG=myapp:* npm start

Windows için bu komutu kullanın:

- set DEBUG=myapp:* & npm start

(NOT: Baştaki DEBUG kısmı her zaman gerekli değildir. Ör: **npm start** direkt projeyi çalıştırır.)

Uygulamaya erişmek için tarayıcınızda http://localhost:3000/ adresini ziyaret edin.

EXPRESS FILE STRUCTURE

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
```

Burada oluşturulan dizin yapısı, Express uygulamasını yapılandırabileceğiniz birçok seçenekten sadece birisidir. İhtiyacınıza en uygun şekilde bu yapıyı kullanabilir ya da düzenleyebilirsiniz.

EXPRESS: ASSIGNING PATH

Basit yol atama

Yol atama, bir uygulamanın HTTP methodu ile sunucuya (GET, POST gibi) gelen isteğe ne şekilde cevap vereceğini tanımladığımız kısımdır.

Gelen istekteki yol, tanımlanan yol ile eşleştiğinde ilgili fonksiyonlar ile kullanıcıya ne dönüleceği kararlaştırılır.

Bir yol tanımlamak için genel yaklaşım şöyledir.:

```
app.METHOD(PATH, HANDLER)
```

Burada:

•app, express objesinin referansıdır.
•METHOD, HTTP metodunun türü (Ör: Get yada POST yada PUT yada DELETE vs..).
•PATH, sunucuda tanımlanmış yol.
•HANDLER, talep edilen yol eşleştiğinde çalışacak fonksiyon.

EXPRESS: PATH EXAMPLE

- Kullanıcının **GET** isteğine Merhaba Dünya! ile cevap vermek için:

```
app.get('/', (req, res) => { res.send('Merhaba Dünya!') });
```

- Kök dizine (/) gelen **POST** isteğine bir cevap verin:

```
app.post('/', (req, res) => { res.send('POST isteği yapıldı!') }) ;
```

- /user yoluna gelen **PUT** isteği:

```
app.put('/iletisim', (req, res) => { res.send('/iletisim adresinde bir PUT isteği') }) ;
```

- /user yoluna gelen **DELETE** isteği:

```
app.delete('/ iletisim ', (req, res) => { res.send('/iletisim adresinde bir DELETE isteği') }) ;
```

EXPRESS: PATH EXAMPLE

- Kullanıcının **GET** isteğine Merhaba Dünya! ile cevap vermek için:

```
const express = require('express’)
const app = express()
const port = 3000
app.get('/', (req, res) => {
        res.send('Merhaba Dünya!’)
})

app.listen(port, () => {
        console.log(`Example app listening on port ${port}`)
})
```

EXPRESS: SERVING STATIC FILES

Görseller, CSS dosyaları ve JavaScript dosyaları gibi statik dosyaları sunmak için, Express'te bulunan **express.static** metodunu kullanabilirsiniz. Birden fazla dizini statik olarak tanımlayabilirsiniz. Bu açıdan kısıtlama yoktur.

Genel kullanım: express.static(root, [options])

Örnek:

```
app.use(express.static('public'))
```

Bu sayede aşağıda yazılı static dosyalara ön yüzden erişim sağlanır

```
http://localhost:3000/images/kedi.jpg
http://localhost:3000/css/mystyle.css
http://localhost:3000/js/jquery.js
```

UYARI: Express statik içeriği kullanıcıya sunarken statik olarak tanımlanmış dizinin altında yer alan dosyalara ve dizinlere erişim verir(bknz public dosyası), bu yüzden uygulamanın kök dizinine göre dosyaların sunulmadığına dikkat ediniz.

EXPRESS EXAMPLE APPS

Express açık kaynak kodlu ve bir topluluğu olan kütüphane/şablon uygulamasıdır. Yine bu topluluk tarafından güncel tutulan örnek uygulamaların yer aldığı web sayfasından ihtiyaçlarınıza uygun olanı seçebilirsiniz.

https://expressjs.com/en/starter/examples.html

- auth - Authentication with login and password
- content-negotiation - HTTP content negotiation
- cookie-sessions - Working with cookie-based sessions
- cookies - Working with cookies
- downloads - Transferring files to client
- ejs - Working with Embedded JavaScript templating (ejs)
- error-pages - Creating error pages
- error - Working with error middleware
- hello-world - Simple request handler
- markdown - Markdown as template engine
- multi-router - Working with multiple Express routers
- mvc - MVC-style controllers with online and redis packages
- params - Working with route parameters
- resource - Multiple HTTP operations on the same resource
- route-map - Organizing routes using a map
- route-middleware - Working with route middleware
- route-separation - Organizing routes per each resource
- search - Search API
- session - User sessions
- static-files - Serving static files
- vhost - Working with virtual hosts
- view-constructor - Rendering views dynamically
- view-locals - Saving data in request object between middleware calls
- web-service - Simple API service

EXPRESS: EXAMPLE APP

Using Cookies

```
var cookieSession = require('cookie-session');
var express = require('express');
var app = module.exports = express();

// Add req.session cookie support
app.use(cookieSession({ secret: 'wedding' }));

const songParts = ['Dört mumdur, on dört mumdur.', 'Bana bir bade doldur.',
  'Bu ne güzel düğündür.', 'Ha ninnah.'
];

// Handle GET request
app.get('/', function (req, res) {
  req.session.count = (req.session.count || 0) + 1;
  let response;
  if (req.session.count === 1) {
    response = 'Bir mumdur.';
  } else if (req.session.count === 2) {
    response = 'İki mumdur.';
  } else if (req.session.count === 3) {
    response = 'Üç mumdur.';
  } else {
    const partIndex = (req.session.count - 4) % songParts.length;
    response = songParts[partIndex];
  }
  res.send(response + '\n');
});

if (!module.parent) {
  app.listen(3000, () => {
    console.log('Express started on port 3000');
  });
}
```

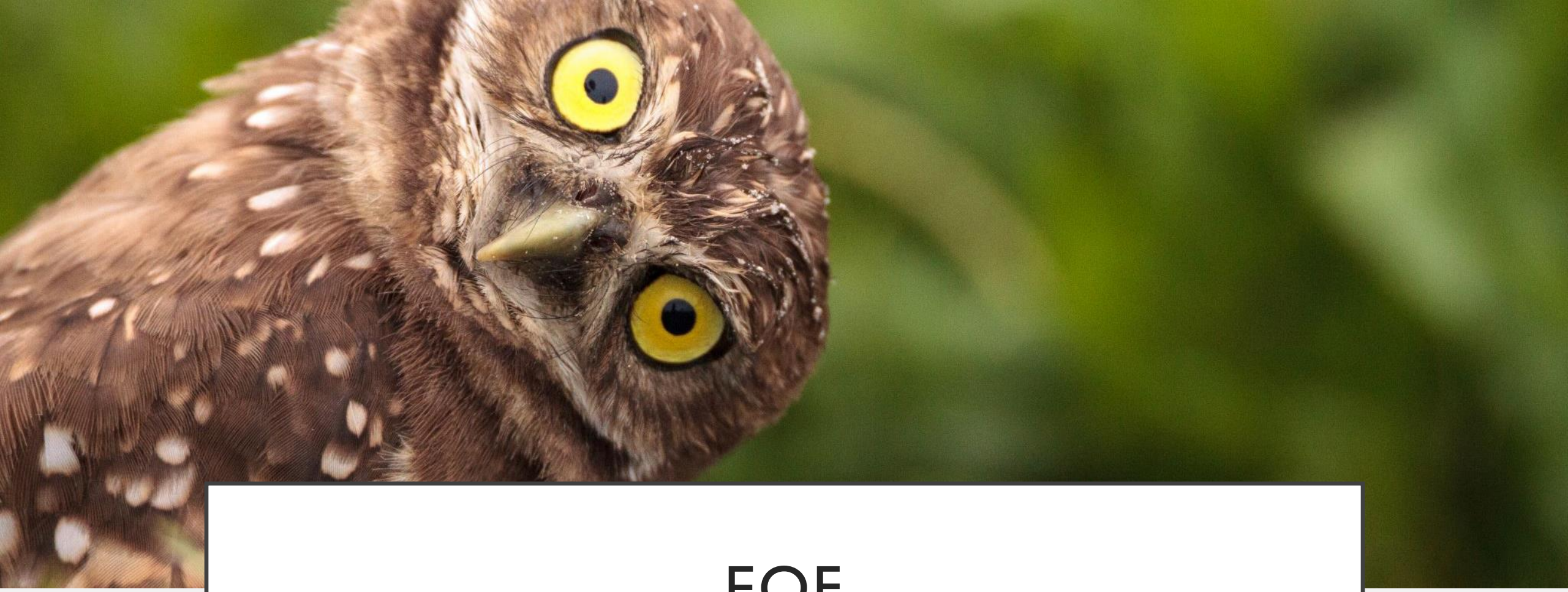
EXPRESS RESOURCES

NODE JS VE EXPRESS ile ilgili Türkçe kaynak arayan ve Veritabanı entegrasyonu ile ilgili çalışma yapmak isteyenler için aşağıdaki eğitimler ve kaynaklardan faydalanılması önerilir.

https://www.yusufsezer.com.tr/node-js-dersleri/

https://www.youtube.com/watch?v=UwQBgXpHs1s&t=3150s&ab_channel=BerkantKAYA

https://www.btkakademi.gov.tr/portal/course/node-js-ile-web-programlama-14301



EOF

REFERENCES/CREDITS

1. Stepp M, Miller J, Kirst V. Web Programming Step by Step. Step by Step Publishing; 2012.
2. CS193X Web Programming Fundamentals Course Slides at Stanford Uni by Victoria Kirst
3. https://www.w3schools.com/nodejs/
4. https://nodejs.org/en
5. https://expressjs.com/tr/