Veb programiranje

Marko Arsenovic
Srdjan Sladojevic

Uvod u Express Framework

Node.js – Express Framework

- Express predstavja minimalni fleksibilni Node.js framework za razvoj single-page i multi page veb i mobilnih aplikacija.
- Middleware bazirani dizajn
- “Unopinionated” – konfigurabilan i komplementan sa drugim bibliotekama
- Inspirisan Sinatra framework-om, po dizajnu
- MEAN = Mongoose Express Angular Node
- Stack u kome se Express najcesce koristi
- Dostupan kao i framework mean.io

Node.js – Express Framework

- Glavne odlike Express framework-a:
  - Omogucuje integrisanje middleware-a za odgovor na HTTP zahteve
  - Definisanje ruting tabele radi implementacije raznih akcija u zavisnosti od URL-a i HTTP metoda
  - Dinamicko renderovanje HTML stana bazirano na prosledjivanju argumenata templejtu

Node.js – Express Framework

- Pre

```
if (page == '/') {
res.end('Welcome !');
} else if (page == '/today') {
res.end('Today :' + new Date());
} else if (page == '/about') {
res.end('This web application runs on Node.js');
}
```

Posle

```
var app = require('express')();
var params = require('express-params');
params.extend(app);

app.get('/', function(req, res) {
res.end('Welcome !');
});

app.get('/today', function(req, res) {
res.end('Today :' + new Date());
});

app.get('/about', function(req, res) {
res.end('This web application runs on Node.js with Express');
});
app.listen(1337);
```

Node.js – Express Framework

- Middleware
- Funkcije koje se pozivaju pre nego što se pozove request handler
- Ove funkcije se pozivaju u redosledu u kom su middleware-i dodati (middleware stack)

Node.js – Express Framework

- Instalacija

```
C:\WINDOWS\system32\cmd.exe
E:\NodePrimeri> npm install express --save
```

- Pored express modula najcesce se koriste i sledeci middleware-i :
  - body-parse -> middleware za rukovanje JSON, Text, URL enkodovanim i sirovim podacima

```
E:\NodePrimeri>npm install body-parser --save
```

  - coockie-parse parsira Cookie heder I postavlja req.cookies sa key-value objektima

```
E:\NodePrimeri>npm install cookie-parser --save
```

  - multer -> middleware za rukovanje multipart/form-data

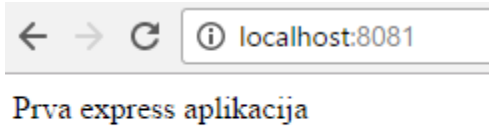
```
E:\NodePrimeri>npm install multer --save
```

Node.js – Express Framework

- Prva aplikacija

server.js

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
    res.send('Prva express aplikacija');
})

var server = app.listen(8081, function () {
    var host = server.address().address
    var port = server.address().port

    console.log("Example app listening at http://%s:%s", host, port)
})
```



Node.js – Express Framework

- Prva aplikacija
- Moze i pomocu express-generator:

  npm install -g express-generator
- Generisanje aplikacije sa Jade templating-om:

  express mojaPrvaAplikacija
- Instaliranje dependencies:

  cd mojaPrvaAplikacija

  npm install

Node.js – Express Framework

- Express aplikacije koriste callback funkcije cisi su parametri request i response objekti

```
app.get('/', function (req, res) {
  // --
})
```

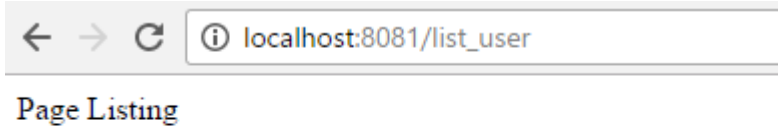
- Request objekat predstavlja HTTP zahtev i poseduje propertie za query string, pararametre, body, HTTP hedere…
- Response objekat predstavlja HTTP odgovor koji Express aplikacija salje kao odgovor na zahtev

Node.js – Express Framework

server.js

```
var express = require('express');
var app = express();

// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})

// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})

// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})

// This responds a GET request for abcd, abxcd, ab123cd, and so on
app.get('/ab*cd', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})

var server = app.listen(8081, function () {

   var host = server.address().address
   var port = server.address().port

   console.log("Example app listening at http://%s:%s", host, port)
})
```

- Rutiranje
- Rutiranje se odnosi na to kako aplikacija odgovra na klijentske zahteve u zavisnotsti od endpoint-a, HTTP zahteva (GET, POST,…), URI-a…



Node.js – Express Framework

- Rutiranje
- Uvezivanje ruta

```
1   app.route('/book')
2   .get(function(req, res) {
3   res.send('Get a random book');
4   })
5   .post(function(req, res) {
6   res.send('Add a book');
7   })
8   .put(function(req, res) {
9   res.send('Update the book');
10  });
```

- Rutiranje
- Rukovanje obradom
- Vise callback-a preko next() funkcije

```
1  app.get('/example/b', function (req, res, next) {
2  console.log('response will be sent by the next function ...');
3  next();
4  }, function (req, res) {
5  res.send('Hello from B!');
6  });
7
```

Node.js – Express Framework

- Rutiranje
- Imenovani parametri – pristup preko req.params

server.js

```
1  router.get('/:user', function(req, res) {
2    res.send("Resource for user " + req.params.user);
3  });
```

- Moze da se implementira biznis logika nad parametrima

server.js

```
1  app.param('username', function (request, response, next, username) {
2    // ... Upit ka bazi
3    // ... Uskladistiti user object iz baze u req object
4    req.user = user;
5    return next();
6  });
7
8
9  app.get('/users/:username', function(request, response, next) {
10   //... Raditi nesto sa  req.user
11   return res.render(req.user);
12 });
```

Node.js – Express Framework

- Error handling middleware

```
if (app.get('env') === 'development') {

  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
        message: err.message,
        error: err
    });
  });

}
```

server.js

```
var express = require('express');
var app = express();
var router = express.Router();

router.get('/',function(req,res) {
  throw new Error("Samo da testiramo error handling!");
  res.send("Home page");
});

app.use('/',router);

app.use(function(err,req,res,next) {
  console.log(err);
  res.status(500).send({"Error" : err.message});
});

app.listen(3000);
```

localhost:3000

```
{"Error":"Samo da testiramo error handling!"}
```

```
:\NodePrimeri>node server.js
rror: Samo da testiramo error handling!
    at E:\NodePrimeri\server.js:6:9
    at Layer.handle [as handle_request] (E:\NodePrimeri\node_modules\express\lib\router\layer.js:95:5)
    at next (E:\NodePrimeri\node_modules\express\lib\router\route.js:137:13)
    at Route.dispatch (E:\NodePrimeri\node_modules\express\lib\router\route.js:112:3)
    at Layer.handle [as handle_request] (E:\NodePrimeri\node_modules\express\lib\router\layer.js:95:5)
    at E:\NodePrimeri\node_modules\express\lib\router\index.js:281:22
    at Function.process_params (E:\NodePrimeri\node_modules\express\lib\router\index.js:335:12)
    at next (E:\NodePrimeri\node_modules\express\lib\router\index.js:275:10)
    at Function.handle (E:\NodePrimeri\node_modules\express\lib\router\index.js:174:3)
    at router (E:\NodePrimeri\node_modules\express\lib\router\index.js:47:12)
```

Node.js – Express Framework

- Staticki resursi
- Ugradjeni midlver express.static za pristupanje statickim fajlovma: slike, CSS, JavaScript…
- Primer

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

- Aplikacija bi koristila staticki direktorijum pod nazivom public u root direktorijumu aplikacije
- Ograničava pristup statičkom sadržaju na folder koji je navedene (nije moguće pristupit nadfolderu sa ../ u putanji)

Node.js – Express Framework

- Odgovori na zahteve

server.js

```
router.get('/', function(req, res) {
  //1.html
  res.send('<h1>Hello World</h1>');

  //2.json
  //res.send({id: '1', username: 'johndoe', name: 'John Doe'});

  //3.status code
  //res.send(404);
  //res.send(500, "Server error");

});
```

server.js

```
//redirect i location

//Redirect with HTTP 302 Found
res.redirect('/pages/example1');

//Redirect with custom HTTP status code
res.redirect(301, 'http://www.example.com/')
```

server.js

```
//response header polja - setters / getters (case-insensitive)

res.set('Content-type', 'text/plain');
var contentType = res.get('Content-type');
```

Node.js – Express Framework

- GET zahtev

server.js | index.html

```html
<html>
    <body>

        <form action = "http://127.0.0.1:8081/process_get" method = "GET">
            First Name: <input type = "text" name = "first_name">  <br>
            Last Name: <input type = "text" name = "last_name">
            <input type = "submit" value = "Submit">
        </form>

    </body>
</html>
```

server.js | index.html

```js
var express = require('express');
var app = express();

app.use(express.static('public'));
app.get('/index.htm', function (req, res) {
    res.sendFile( __dirname + "/" + "index.htm" );
})

app.get('/process_get', function (req, res) {
    // Prepare output in JSON format
    response = {
        first_name:req.query.first_name,
        last_name:req.query.last_name
    };
    console.log(response);
    res.end(JSON.stringify(response));
})

var server = app.listen(8081, function () {
    var host = server.address().address
    var port = server.address().port
    console.log("Example app listening at http://%s:%s", host, port)

})
```

localhost:8081/index.htm

First Name: Petar
Last Name: Petrovic
Submit

127.0.0.1:8081/process_get?first_name=Petar&last_name=Petrc

```
{"first_name":"Petar","last_name":"Petrovic"}
```

Name

- node_modules
- public
- index.htm
- server.js

Node.js – Express Framework

- Parsiranje tela zahteva i preuzimanje parametara:
  - Parsiranje se vrši ukoliko je content-type application/JSON ili application/x-www-form-urlencoded
  - Kreiranje JavaScript objekta koji reprezentuje zahtev
- Middleware body-parser

Node.js – Express Framework

- POST zahtev

server.js index.htm

```html
<html>
    <body>

        <form action = "http://127.0.0.1:8081/process_post" method = "POST">
            First Name: <input type = "text" name = "first_name"> <br>
            Last Name: <input type = "text" name = "last_name">
            <input type = "submit" value = "Submit">
        </form>

    </body>
</html>
```

server.js index.htm

```javascript
var express = require('express');
var app = express();
var bodyParser = require('body-parser');

// Create application/x-www-form-urlencoded parser
var urlencodedParser = bodyParser.urlencoded({ extended: false })

app.use(express.static('public'));
app.get('/index.htm', function (req, res) {
   res.sendFile( __dirname + "/" + "index.htm" );
})

app.post('/process_post', urlencodedParser, function (req, res) {
   // Prepare output in JSON format
   response = {
      first_name:req.body.first_name,
      last_name:req.body.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port

   console.log("Example app listening at http://%s:%s", host, port)

})
```

localhost:8081/index.htm

First Name: Petar
Last Name: Petrovic Submit

127.0.0.1:8081/process_post

{"first_name":"Petar","last_name":"Petrovic"}

Node.js – Express Framework

- Primer Upload fajla

server.js | index.htm

```html
<form enctype="multipart/form-data" action="/upload" method="post">
  <input type="file" name="photo" />
  <input type="submit" value="Upload Image" name="submit" />
</form>
```

127.0.0.1:5000

Choose File coton3.jpg Upload Image

127.0.0.1:5000/upload

Your File Uploaded

PC › PODACI (E:) › NodePrimeri › uploads

coton3.jpg

server.js | index.htm

```js
var express = require('express');
var multer = require('multer');
var app = express();
var port = 5000;

app.set('port', port);

var storage = multer.diskStorage({
  destination: function (request, file, callback) {
    callback(null, __dirname +'/'+'/uploads');
  },
  filename: function (request, file, callback) {
    console.log(file);
    callback(null, file.originalname)
  }
});
var upload = multer({storage: storage}).single('photo');

app.get('/', function(resuest, response) {
  response.sendFile(__dirname +'/'+ 'index.htm');
});


app.post('/upload', function(request, response) {
  upload(request, response, function(err) {
  if(err) {
    console.log(err.message);
    return;
  }
  console.log(request.file);
  response.end('Your File Uploaded');
  console.log('Photo Uploaded');
  })
});

var server = app.listen(port, function () {
  console.log('Listening on port ' + server.address().port)
});
```

Node.js – Express Framework

- Templating
- Template engine-i mogu biti ugradjeni u Express: Jade, Mustache, Handlebars, EJS…

C:\WINDOWS\system32\cmd.exe

```
E:\NodePrimeri>npm install express-handlebars
```

Node.js – Express Framework

Name

node_modules

views

server.js

PODACI (E:) › NodePrimeri › views

Name

home.hbs

- Templating
- Handlebars primer

server.js home.hbs

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Example App - Home</title>
    </head>
    <body>
    <!-- Uses built-in `if` helper. -->
    {{#if showTitle}}
    <h1>Home</h1>
    {{/if}}
    <!-- Calls `foo` helper, overridden at render-level. -->
    <p>{{foo}}</p>
    <!-- Calls `bar` helper, defined at instance-level. -->
    <p>{{bar}}</p>
    </body>
</html>
```

localhost:3000

Home

foo.

BAR!

server.js home.hbs

```javascript
var express = require('express');
var exphbs = require('express-handlebars');
var app = express();
var hbs = exphbs.create({
    // U helepre se smesta logika, built in su with, if, each itd.
    helpers: {
    foo: function () { return 'FOO!'; },
    bar: function () { return 'BAR!'; }
    }
});
app.engine('hbs', hbs.engine);
app.set('view engine', 'hbs');
app.set('views', __dirname + '/views');
app.get('/', function (req, res) {
    res.render('home', {
        showTitle: true,
        // Override `foo` helpera-a samo za ovo renderovanje
        helpers: {
            foo: function () { return 'foo.'; }
        }
    });
});
app.listen(3000);
```

Node.js – Express Framework REST

- REST (Representational State Transfer) -> softverski stil arhitekture veba
- Skup principa koji definise kako bi veb standardi poput HTTP I URI-a trebali da budu iskorisceni
- Roy Thomas Fielding - doktorska disertacija *Stilovi arhitekture i dizajn mrežno baziranih softvera arhitekture*
- REST ogranicenja:
  - Client – Server
  - Stateless
  - Cache
  - Uniform interface
  - Code on demand

Node.js – Express Framework REST

PRIMER

users.json | server.js

```
[{
    "name": "Pera",
    "password": "pass1",
    "profession": "QA",
    "id": 1
}, {
    "name": "Marica",
    "password": "pass2",
    "profession": "Developer",
    "id": 2
}, {
    "name": "Jovic",
    "password": "pass3",
    "profession": "Manager",
    "id": 3
}]
```

REST Endpoint-i

	URI	HTTP Method	POST body	Result
1	listUsers	GET	empty	Prikaz svih user-a
2	addUser	POST	JSON String	Dodavanje novog
3	deleteUser/:id	DELETE	JSON String	Brisanje postojeceg
4	:id	GET	empty	Prikaz detalja

node_modules
server.js
users.json

Node.js – Express Framework REST

GET - listUsers

users.json | server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/listUsers', function (req, res) {
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      console.log( data );
      res.end( data );
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)

})
```

Node.js – Express Framework REST

- POST - addUser

users.json | server.js

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
var fs = require("fs");
// parse application/json
app.use(bodyParser.json())

var count

app.post('/addUser', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       data.push(req.body);
       console.log( req.body );
       var jsonData = JSON.stringify(data);
       fs.writeFile(__dirname + "/" + "users.json", jsonData);
       res.end(jsonData);
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)

})
```

Postman

POST http://localhost:8081/addUser Params Send Save

```
{
    "name" : "Ana",
    "password" : "pass4",
    "profession" : "Boss",
    "id": 4
}
```

Body Cookies Headers (4) Tests Status: 200 OK Time: 111 ms

Pretty Raw Preview Text


```
[{"name":"Pera","password":"pass1","profession":"QA","id":1},{"name":"Marica","password":"pass2","profession":"Developer","id":2},{"name":"Jovic"
    ,"password":"pass3","profession":"Manager","id":3},{"name":"Ana","password":"pass4","profession":"Boss","id":4}]
```


Node.js – Express Framework REST

- GET :id

users.json | server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/:id', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       users = JSON.parse(data);
       var user;
       for(var i = 0; i < users.length; i++) {
           if(users[i].id ==  req.params.id){
               user = users[i];
               break;
           }
        }
       console.log( user );
       res.end( JSON.stringify(user));
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)

})
```

localhost:8081/1

{"name":"Pera","password":"pass1","profession":"QA","id":1}

Node.js – Express Framework REST

- DELETE deleteUser/:id

server.js | users.json

```
var express = require('express');
var app = express();
var fs = require("fs");

var id = 2;

app.delete('/deleteUser/:id', function (req, res) {

   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
      var users = JSON.parse( data );
      console.log(users);
      for(var i = 0; i< users.length;i++){
         if(users[i].id == req.params.id){
            users.splice(i,1);
            break;
         }
      }
      var jsonData =  JSON.stringify(users);
      fs.writeFile(__dirname + "/" + "users.json", jsonData);
      res.end(jsonData);
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)

})
```

DELETE http://localhost:8081/deleteUser/2 Params Send

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

Body Cookies Headers (4) Tests Status: 200 OK

Pretty Raw Preview Text

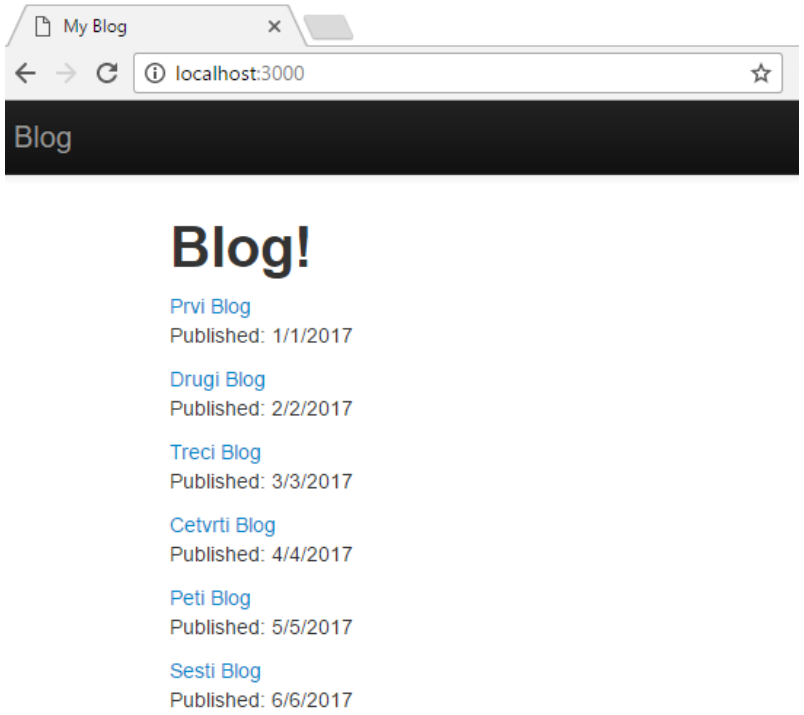

```
[{"name":"Perica","password":"pass1","profession":"QA","id":1},{"name":"Jovica","password":"pass3","profession":"Manager","id":3},{"name":"Ana"
,"password":"pass4","profession":"Boss","id":4}]
```


server.js | users.json


```
[
   {
   "name":"Perica",
   "password":"pass1",
   "profession":"QA",
   "id":1
   },
   {"name":"Marica",
   "password":"pass2",
   "profession":"Developer",
   "id":2
   },
   {"name":"Jovica",
   "password":"pass3",
   "profession":"Manager",
   "id":3
   },
   {"name":"Ana",
   "password":"pass4",
   "profession":"Boss",
   "id":4
}
]
```


Node.js – Express Framework

- ZADATAK
- Blog aplikacija
- Na postojecoj aplikaciji dodati:
  - Skladistenje blogova fajl sistem – blogs.json
  - Dodavanje novog, brisanje starog bloga

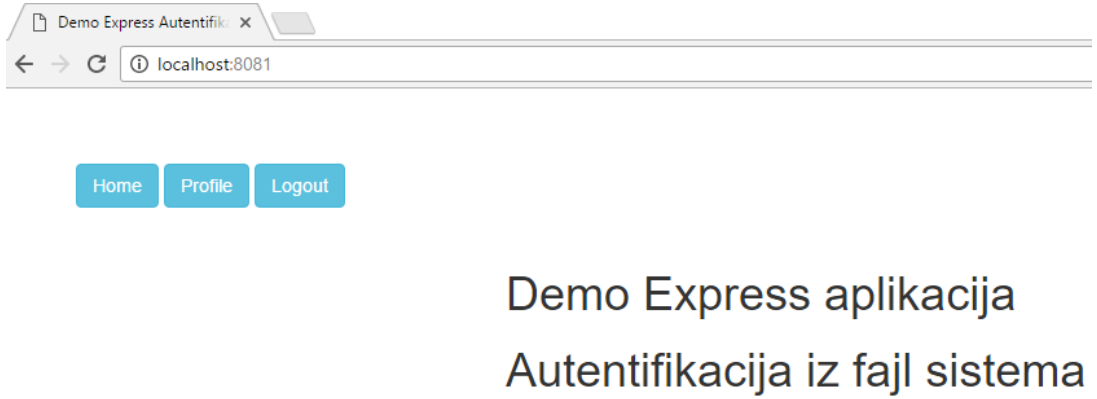


Node.js – Express Framework

- ZADATAK - DODATAK
- Blog aplikacija
- Na postojecoj aplikaciji dodati mogucnost autentifikacije korisnika
- Baza korisnika je u fajl sistemu
- Admin ima prvo da dodaje novi blog ili da brise stari
- Ostali imaju samo parvo citanja ostalih blogova

Node.js – Express Framework

- DODATAK
- Primer autentifikacije koriscenjem Passport middleware-a

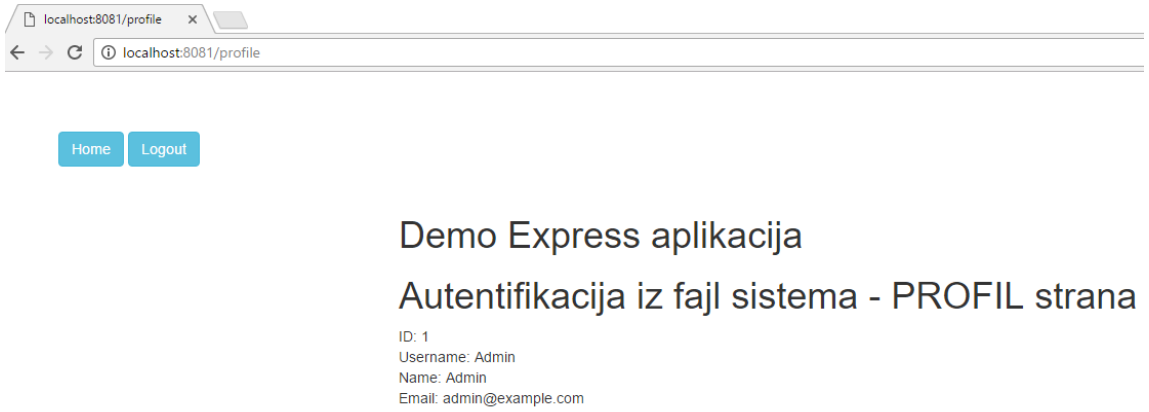


- baza je u fajl sistemu

```
var records = [
    { id: 1, username: 'Admin', password: 'admin', displayName: 'Admin', emails: [ { value: 'admin@example.com' } ] },
    { id: 2, username: 'User', password: 'user', displayName: 'User', emails: [ { value: 'user@example.com' } ]}
];
```

Node.js – Express Framework

- DODATAK
- Primer autentifikacije koriscenjem Passport middleware-a



Demo Express aplikacija

Autentifikacija iz fajl sistema - ADMIN strana

Node.js – Express Framework

- Korisni linkovi:
- http://www.vanmeegern.de/fileadmin/user_upload/PDF/Web_Development_with_Node_Express.pdf
- https://hackerstribe.com/wp-content/uploads/2016/04/Node.js-Express-in-Action.pdf