

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width,
    initial-scale=1.0">
  <link rel="stylesheet" type="text/css"
    href="styles/main.css">
  <script src="scripts/main.js"></script>
</head>
<body></body>
</html>
<figure>, <ul> - neuredena lista
<tr> -> red, <td> -> stupac, <ol> - uredena lista
<a href="http://www.fer.hr/index.html">fer</a>
<a href="#b2" title="1. poglavlje">poglavlje</a>
<h3 id="b2">Skoci na ovo poglavlje</h3>
<hr /> <!-- horz crta-->

<form target="_self" action="/processForm.php" method="GET">
  <!-- target: self unutar istog, blank unutar novog -->
  <label>Korisnicko ime: <input type="text"
    name="username" value="enter your username"
    size="30"></label><br/>
  <label>Lozinka: <input type="password"
    name="password"
    maxlength="30" required></label><br/>
  <label>Skriven input <input type="text" hidden readonly></label>

<fieldset> <!-- radio-button -> izbor jednog -->
  <legend>Uloga</legend>
  <label><input type="radio" name="role"
    value="admin">Administrator</label><br/>
  <input type="radio" id="user" name="role" value="user" checked>
  <label for="user">Korisnik</label>
</fieldset>
<fieldset> <!-- checkbox -> bilo koji broj -->
  <legend>Dodatne opcije</legend>
  <label><input type="checkbox"
    name="stakla" value="da" checked>Stakla</label><br/>
  <label><input type="checkbox"
    name="felge" value="da">Felge</label>
</fieldset>
<select name="padajuca_lista" size="1">
  <!-- multiple za selectanje vise -->
  <option value="opcija" selected>Opcija</option>
  <option value="opcija">Opcija2</option>
</select><br/>
<input type="submit" value="Submit">
<input type="reset" value="Odustani">
</form>
  <!--
    greske kodiranja sadržaja:
    ponovljeno ime elementa, atribut disabled, atribut nije definiran
    application/x-www-form-urlencoded -> cust=Pero+Peri%C4
    text/plain -> za developere -> cust=Pero address=Ulica
  -->

```

Korisnicko ime:

Lozinka:

Skriven input

Uloga

☐ Administrator

☒ Korisnik

Dodatne opcije

☒ Stakla

☐ Felge

Opcija ▼

Submit Odustani

```

Jednostavni:
  * => sve
  h1, li => svi h1 ili li
  li.c1 => svi li s klasom c1
Atributni:
  li[id="z2"] => svi li s id=z2
Kombinirani:
  div span => svi span unutar div
  div > span => neposredna djeca span roditelja div
  div+span => prvi span nakon diva, ista razina
  div~span => isto kao +, ali sve elemente nakon diva iste razine
Pseudoklasa:
  div:hover => na hover misa
  li:first-child => li koji je prvo dijete roditelja
  input:required => svi required inputi
Pseudoelementi:
  p::before, p::after {content: "\\"; } => ubaci " prije i poslije paragrafa
  p::first-letter, p::first-line
  p::selection => dio elementa koji je odabrao korisnik
  inline => 1000
  #id => 100
  .class, :pseudo-class, [attribute] => 10
  <TAG>, ::pseudo-element => 1
  <body>, * => 0
  border: inherit => uzmi od roditelja
  border: initial => iskljuci sve, uzmi od browsera
  border: unset => inherit ako ima matching value, initial inace
Neki css elementi: font-family, font-weight:bold,
  font-size, font-style:italic, text-decoration:underline/none
  text-align, text-indent, letter-spacing, line-height
  background-image: url('./images/x.jpg'),
  background- color: green
  background-repeat: no-repeat, repeat
  box-sizing: border-box
  display: block/inline/inline-block
  padding, border, margin
  margin:auto => centriranje
  max-width, width,vw,vh,vmin,vmax
  position: relative/absolute/fixed
Važniji globalni atributi: id,class,
  lang,title,style
*****JS-MI*****

```

```

var x = 123e-5; let y = "string";
const z = 'string';
// "1"+"2"="12", Number(1)+Number(2)=3
let exp = 2**3 // 2^3 = 8
// strings \\
let s = "he said: \"xd\"";
let len = s.length;
let index = s.indexOf("xd"); // .lastIndexOf()
let newString = s.slice(1,2); // [od,do),e
let numToString = (123).toString();
let stringToNum = Number(numToString);
// functions \\
function f1(x="default value") {
  console.log(x);
}
let f2 = function(x) {console.log(x);}
let f3 = (x) => console.log(x);
// arrays \\
let arr = [1,2,3,5,6,1,7,8];
let last = arr.pop(); // makne i returna zadnji
arr.push(4); // append na kraj
let first = arr.shift(); // makne i returna prvi
arr.unshift(5); // ubaci na pocetak
arr.splice(1,2); // brise 2 elementa od indeksa 1
let arr2 = [6, 7];
let arr3 = arr.concat(arr); // spoji
JSON.stringify - pretvara objekte u JSON format
JSON.parse - učitava objekte iz JSON formata

```

```

ZI
rem -> u odnosu na root (default je 16px)
em -> u odnosu na roditelja
<div style="font-size: 30px;">
  <span style="font-size: 1em;">em</span>
  <span style="font-size: 1rem;">rem</span>
</div> <!-- em=30px, rem=16px -->

```

```

*****
arr4 = arr3.slice(1,3); //indexi od 0, 1 element,2 element, 3 ne
console.log(arr4.includes(6)); //vraca bool
arr.sort( function(a,b){return b - a } ); // sort sa komparatorom
(desc)
arr.reverse();
arrEven = arr.filter( (x) => x%2==0);
for (let element of arr) {console.log(element);} // elementi
for (let index in arr) {console.log(index);} // indeksi
// classes, objects \\
class Person {
  lastName = "Doe";
  age = 50;
  constructor(firstNameValue) {this.firstName =
firstNameValue;}
  get lastName() {return this.lastName;}
  set lastName(newLastName) {this.lastName =
newLastName}
}
let person1 = new Person("John");
person1.lastName = "Williams";
person1.firstName = "Jake";
for (let value of Object.values(person1)) {console.log(value)}
// try-catch \\
try {
  throw "error";
} catch(err) {
  console.log(err);
} finally {
  var x = 2; }
let mapa=new Map();
mapa.set('mile','car');
mapa.get('mil') //undefined
mapa.has("mile");
mapa.delete("mile");
mapa.clear()

```

```

Booelan
(undef,null,NaN,0,"")
=false
Number(
null,false=0
undefined=NaN
true=1)
null===undef false
null===undef true

```

```

setTimeout( () => console.log(3) , 3000 ); // nakon 3s
let promise = new Promise( (resolve, reject) => {
    setTimeout( () => {
        console.log("nakon 3 sekunde...");
        if (false) {
            resolve("dobro izvršen");
        } else {
            reject("loše izvršen");
        }
    },
    3000);
});
promise.then(
    function(result) {console.log(result);},
    // ako resolve, result = "dobro izvršen"
    function(error) {console.log(error);}
    // ako reject, result = "loše izvršen"
);
promise.catch(
    function(error) {console.log(error);} // samo ako reject
)
promise.catch(
    function(error){console.log(error);}
).then(
    function(result){console.log("Resolve:"+result)},
    function(result){console.log("Reject:"+result)}
); // catch će uhvatiti error, u then se će pozvati prva funkcija s result = undefined
//Fetch\
let promise2 = fetch("https://web1lab2.azurewebsites.net/products?categoryId=1");
promise2.then( // obrađuje se promise od fetcha
    function(response) {
        if (!response.ok) { throw new Error("Cannot load"); }
        else { return response.json(); } // novo obecanje response.json()
    },
    function(error) { throw error; }
).then( // obrađuje se promise od response.json
    function(response) { console.log("Loaded JSON"); }
).catch( // catch hvata grešku u bilo kojem promiseu
    function(error) { console.log(error); }
)
//LoadJson\
async function LoadJSON() { // funkcija se izvodi asinkrono
    let promise = await fetch("https://web/categoryId=1");
    // unutar funkcije, await se izvodi sinkrono (ostatak funkcije čeka)
    if (!promise.ok) { throw new Error ("Cannot load"); }
    else { var jsonContents = await promise.json(); }
    console.log(jsonContents);
}
LoadJSON().catch(
    (error) => {console.log(error);}
)
***** prez 8 *****
GET{metoda} / predmet / rppzwpw HTTP{oznaka resursa}/1.1{oznaka protokola}
Host: www.fer.hr {ime poslužitelja}
HTTP - (hypermedia) prijenos u formatima -> html,meta-data,chunk,
Media Type -> text/html,image/jpeg,video/quickTime,application/javascript

logo.png?
Browser -----> www.fer.hr
Content-type image/jpeg
Content-length:1399
<-----

MME Type - (tip/podtip) -> application/javascript, application/json, text/plain
Ciklus zahtjev-odgovor= jedna konverzacija
HTTPS port(443) --> HTTPS --> TLS/SSL --> TCP --> IP
Uspostava komunikacije TLS - faza rukovanja(dogovor parametara),
faza komunikacije(ključ za šifriranje poruka)
Tijek komunikacije Server <-> Client
<- šalje zahtjev , -> odgovara certifikatom, <- provjerava certifikat, generira ključ sjednice
, šalje ključ šifriran javnim ključem, -> dešifrirati ključ sjednice, <-> koriste ključ sjednice

```

```

Primjena HTTP - Cloud Computing, Rest, www, WOT
URI - uniformni (struktura zapisa)
- identifikator (informacija koja omogućuju razlikovanje resursa)
- resurs (informacijski izvor)
URL,URN - podskup od URI
URN -jedinstvenost i trajnost identifikacije
pr. urn:ietf:rfc:2616
URL - sadrži informaciju o lokaciji
pr. http://www.ietf.org/rfc/rfc.txt
Primjeri URI-a (http://www.ietf.org/rfc/rfc2396.txt,mailto:John.Doe@example.com
--> apsolutni (puno ime web adrese,www.fer.hr)
--> relativni (skraćeno, npr localhost)

Analiza URI-a
http:{shema,nacin pristupa resursu(HTTP)}/{www.fer.unizg.hr{host name(ip adresa ili ime)}/{
|kako| |gdje|
predmet/rppzwpw{put resursa}
|sto se dohvaća|
shema:(http,ftp,urn,file)// <autoritet> <put {{predmet}> ? <upit {web=prag}> {put,upit isto
neobavezno}
pr. http://www.google.com:81/search?q=web{html#b3 -> skakanje po poglavljima}
<a href="..djelatnost/nastava/intstv.html">Internet stvari </a> (popni se na folder više, spusti na
djelatnost/nastava/ , otvori intstv.html

Poruke HTTP
request reply
Get /pred/web HTTP 1.0 HTTP 1.1 200 OK Pocetni redak
... Content-type Zaglavlja
...
prazan redak
... <!doctype html> <html> tijelo poruke

Metode zahtjeva
GET - dohvaćanje sadržaja, HEAD- dohvaćanje podataka o resursu(nema sadržaja u tijelu za
razliku od GET)
, POST(sign up, comment,burza grupa),PUT,DELETE
HTTP reply - HTTP/1.1 {inacica protokola} 404 {kod} Not Found {opis}
Kod - Informacija ->(100 Continue), Uspjeh -> ( 200 Ok), Preusmjeravanje -> (300 Multiple
Choices, 301 Moved, 302 Found, 304 Not Modified), Pogreska na klijentu -> (400 Bad Request,
401 Unauthorized, 403 Forbidden, 404 Not Found), Pogreska na poslužitelju->(500 Internal server
error, 503 Service not available, 505 http version not supported)
GET koristi link, POST body

opera dns server www.fer.hr | Validacija - može se povući js koji radi validaciju da se
fer.hr? | ne šalje na server
-----> |
<----- |
GET /pred/web | Uvjetni GET -> IF-Match,IF-None-Match,If-Range
-----> |
HTTP 200 OK +index.html |
<----- |
GET css |
-----> |
GET js |
-----> |
<----- |
<----- |

***** prez 9,10 *****
Procesni modeli i protokoli
-> in-process (opasno,ISAPI,Apache Server Api,low usage)
-> poseban proc(sporo,CGI,low usage)
-> poseban proc s pool-om(Fast CGI,PHP)
-> proc s 2 dretve
-> proc s pool-om
-> vanjski proc s pool-om dretvi
Arhitecture
browser <----> server <----> vanjski intrepeter(python)
browser <----> server <----> aplikacijski server(Node.js)
Event Loop -> ako je fja async stavlja se u queue sve dok se sve ne obradi

Versioning -> patch ~version 1.2.3 -> [1.2.3, 1.3.0>
-> minor ^version 1.2.3 -> [1.2.3, 2.0.0>
-> major *version

```

Promises

```
let makePromise=function (x) {
  return new Promise(function (res, rej) {
    try {
      setTimeout(function () {
        res(x);
      }, 1000)
    } catch (err) {
      //handle err
    }
  })
}

let afAll = async function(){
  let sum=0;
  let res = await Promise.all([
    makePromise(getRandomBetw([1,5])).then(function (r1){
      sum+=r1;
    })
    .catch(function (err){
      //handle err
    }),
    makePromise(getRandomBetw([1,5])).then(function (r2){
      sum+=r2;
    }).catch(function (err){
      //handle err
    })
  ])

  let sum=0;
  makePromise(getRandomBetw([1,5]))
  .then(function (r1){
    sum+=parseInt(r1);
    return makePromise(getRandomBetw([1,5]));
  }).then(function (r3){
    sum+=parseInt(r3);
    console.log(`sum is ${sum}`);
  }).catch(function (err){
    //handle err
  })

  let asyF = async function () {
    let r1 = await makePromise(getRandomBetw([1,5]));
    let r2 = await makePromise(getRandomBetw([1,5]));
    let r3 = await makePromise(getRandomBetw([1,5]));
    console.log(`${r1}+${r2}+${r3}=${r1+r2+r2}`);
  }
}
```

***** prez 11, 12 *****

<%= x %> -> x

<%- @x %> -> @x

validacija - provjera ispravnosti podataka

(moze se provesti na: serveru, bazi, klijentu)

-> forma(disabled, maxlength, max, min, step)

-> js (regex, neka fja)

Stanja -> na razini citavog sustava(globalno)

-> na razini korisnika sustava(kosarica)

-> na razini sjednice izmedu korisnika i sustava(login)

Tranzijetna pohrana -> nema trajnog cuvanja stanja

Prezistentna -> trajno cuvanje(pr. sustav i korisnik)

Sjednica -> slijed vremenski omedenih i logicki povezanih transakcija izmedu pojedinog klijenta i poslužitelja

1. pocetak sjednice(zahitjev klijenta prema serveru nakon duljeg vremena neaktivnosti)
2. trajanje sjednice(logicki povezane transakcije izmedu klijenta i servera)
3. zavrsetak sjednice(prestanak rada klijenta)

Identifikator sjednice(session token) -> određuje sjednicu, dodan svakoj transakciji koja pripada sjednici

Prijenost stanja -> hidden field, URL rewriting, cookies

Hidden field -> <input name="naziv" type="hidden" value="SID=abc123">

--> pros - podrzan na svakom browseru, ne moze se onemoguciti, performanse

--> cons - vidljivi kod izvornog koda, prijenos kod svake transakcije, korištenje obrazaca

//index.js\\

//implementacija session-a

router.use(session.sessionManager);

if(req.session.access_counter === undefined) // postavi access_counter koji smo izmislili

req.session.access_counter = 0;

//sessionFER\\

//session record store

let sessionStore = new Map();

//extract sessionID from GET or POST request

let sessionID = (req.query[sIDName] || req.body[sIDName]);

//fetch the session record

let sidRecord = sessionStore.get(sessionID);

if(!sidRecord) {

sidRecord = {id: uuid.v4(), created: Date.now()};

sessionStore.set(sidRecord.id, sidRecord)

//add the session record to the request object

req.session = sidRecord;

//pass the control to the next middleware layer

next();

Url rewriting -> mehanizam oznacavanja sjednica kada cookie nije dostupan

(https://www.fer.unizg.hr/predmet/or?sid=234a3f0cc7)

--> pros - neovisan o browseru, ne moze se onemoguciti na klijentu,jednostavan

--> const - prijenos kroz URI, ogranicena kolicina, manja citljivost, dodatna funkcionalnost

//add sessionID parameter to URL query segment

return function(url) {

let newURL = new URL(url)

newURL.searchParams.append(sIDName, sessionID)

return newURL.toString()

}

Cookies -> mala kolicina slobodno definiranih vrijednost, do 4kB

-> stvara server, sprema klijent

--> domena+put=doseg

--> sadrzaj - ime=vrijednost,obvezno

--> domena - ako nije definirano uzima se od servera, npr www.fer.unizg.hr/predmet/or

--> put - ako nije definiran uzima se dio URI-a,fer.unizg.hr/nastava/or/labosi.html --> /nastava/or

-->rok valjanosti, ogranicenje pristupa,sigurnost prosljedivanja(isto za druge domene) <-- opcionalno

GET /predmet/or

Host www.fer.unizg.hr

Client----->

Set-cookie: sid=mileVOliDisko(sadrzaj);

Path=/nastava/or(put);Domain=www.fer.unizg.hr(domena);

Secure(sigurna veza);HttpOnly(nema lokalnog pristupa);

Expires: Wed, ...(istjece, moze i Max-age=3600)

<-----Server

Uvjeti prosljedivanja cookie-a

1. server pripada domeni (pr. www.fer.unizg.hr(*host-only),fer.unizg.hr,unizg.hr,hr ->da, carnet.hr->ne)
2. sadrzan unutar puta (/nastava/or/labosi,nastava/or->da, nastava/oop-> ne)
3. nije isteko rok trajanja, 4.ako je defirniran secure salje se kroz https(ne http)
5. ako zabranimo, cookie nece bit prosljeden iz druge domene

GET /nastava/or

Host www.fer.unizg.hr

Cookie: sid=abc123

Client----->

HTTP/1.1 OK

Content-type: text/html

...

<-----Server

GET /intranet/or

Host www.fer.unizg.hr

Cookie: sid=abc123

Client----->

Trajni -> definiran rok valjanosti

SameSite

-> none (cookie se salje na drugu domenu)

-> strict(cookie preko druge domene se ne salje, link na nju ne radi)

-> lax

(cookie preko druge domene se ne salje ali radi link na nju)

//cookies.js\\

res.cookie(req.query.name, req.query.value, { path: req.query.path })

res.clearCookie(req.query.name, {path: req.query.path})

//app.js\\

const cookieParser = require('cookie-parser')

//cookie parser middleware

app.use(cookieParser());

//page.js\\

router.get('/', function(req, res, next) {

res.render('page', {

path: req.path,

cookies: req.cookies

});

*****server.js*****

const express = require('express');

const app = express();

const path = require('path');

const pg = require('pg')

const db = require('./db')

const session = require('express-session')

const pgSession =

require('connect-pg-simple')(session)

const router = require('./routes/router');

app.set('views', path.join(__dirname, 'views'));

app.set('view engine', 'ejs');

app.use(express.static

(path.join(__dirname, 'public')));

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

app.use(session({

store: new pgSession({

pool: db.pool,

}),

secret: 'fer-web-lab4',

resave: false,

saveUninitialized: true,

cookie: {maxAge: 24 * 60 * 60 * 1000}

});

app.use('/register', router);

app.listen(3000);

Opis modela : upisujemo mail i password

i tako se registriramo, ispisujemo stare mailove i trenutnog, req.session.user =

{email: req.body.email}; moglo se i

req.session.user=req.body.email ali ovako

mozemo dodati i neki dodatni parametar kojem

mozemo pristupiti preko ejs-a, npr username:

req.body.username

*****MI*****

<body>

<p id="i1" class="c2"> Paragraf </p>

<button onclick="decr()"></button>

<input id="cnt" type="text"

value="5" readonly/>

</body>

function decr(){

let cnt=document.getElementById("cnt");

cnt.value=Number(cnt.value)-1;

let p = document.getElementById("i1");

p.style.backgroundColor = "Red";

p.innerHTML = "promjena teksta";

let newP = document.createElement("p");

body.appendChild(newP);

*****router.js*****

const express = require('express');

const router = express.Router();

const {body,validationResult} = require('express-validator');

const db = require('./db');

router.get('/', async function (req, res, next) {

let rsp = await db.query('SELECT email FROM users');

//router.get('/:id', function(req,res,next) {

//id = parseInt(req.params.id)

// rsp.rows[i] pristup elementima,

// rsp.rows[i].atribut pristup atributu

// www.testovi.com/test/:1?a=b&c=d --> req.query.a -> b, req.query.c -> d

//req.param.id -> 1

res.render('register', {

title: 'Register',

err: undefined,

users: rsp.rows,

user: req.session.user

});

});

router.post('/', [

body('email').trim().isEmail(),

body('pass').trim().isLength({ min:3, max:20 })

//body('employedsince').toInt().isInt({min:1970,max:2021}),

],

async function (req, res, next) {

const errors = validationResult(req);

if (!errors.isEmpty()) {

res.render('register', {

title: 'Register',

err: "Invalid input!",

users: [],

user: req.session.user

});

} else {

try {

await db.query('INSERT INTO

users(email, password) VALUES (\$1, \$2)',

[req.body.email, req.body.pass]);

req.session.user = {email: req.body.email};

res.redirect('/register');

} catch (err) {

console.log(err);

res.render('register', {

title: 'Register',

err: "Database error!",

users: [],

user: req.session.user

});

}

} module.exports = router;

*****register.ejs*****

<html>

<head>

<title> <%= title %> </title>

</head>

<body>

<%- include('partials/header') %>

<form action="/register" method="POST">

<fieldset>

<legend>Registration data</legend>

<div>

<label for="email">Email:</label>

<input type="text" name="email" id="email"

maxlength="20"

minlength="2" size="30">

</div>

<div>

<label for="password">Password:</label>

<input type="text" name="pass" id="password"

maxlength="20"

minlength="2" size="30">

</div>

<div>

<input class="btn" type="submit" value="Submit">

<input class="btn" type="reset" value="Reset">

</div>

<% if (err !== undefined) { %>

<div>

<%= err %>

</div>

<% } %>

</fieldset>

</form>

<div>

Used emails:

<% for (usr of users) { %>

<%= usr.email %>

<% } %>

</div>

<% if (user !== undefined) { %>

<div>

This session:

<%= user.email %>

</div>

<% } %>

</body>

</html>

-----Dodatna Teorija-----

*****Cookies*****

Sigurnost: informacije unutar kolačića su vidljive i čitljive kod prijenosa nesigurnim komunikacijskim kanalom, presretanje(krađa) kolačića, Cross-site scripting(Session cookie theft, Session hijacking), Cross-site request forgery, Cookie poisoning -> third-party cookies su problem privatnosti

Trajnost kolačića: Trajni (persistent) kolačići (imaju definirano vrijeme isteka, brišu se nakon roka isteka valjanosti) I Privremeni (transient) – session cookies (nemaju definirano vrijeme isteka valjanosti, najčešće imaju maxAge, brišu se nakon prestanka rada klijenta)

Brisanje kolačića – “ručno” (korisnik putem opcije u pregledniku), automatsko brisanje istekom roka valjanosti, promjenom sadržaja kolačića

Pristup kolačiću – udaljen (poslužitelj, mehanizmom proslijeđivanja kolačića putem HTTP odgovora) ili lokalni (klijent, programski)

- ako je postavljeno svojstvo HttpOnly nije moguć lokalni pristup kolačiću

- ograničenje proslijeđivanja samo na sigurne komunikacijske kanale (Secure)

KOMUNIKACIJA S BAZOM PODATAKA

Driver održava bazen (pool) N unaprijed otvorenih konekcija s istim korisničkim računom-> CONNECTION POOLING

MVC – MODEL (UserModel s labosa)

- zaduženje: dohvat I manipulacija podacima

- tipično u suradnji s bazom podataka

- JS file

MVC – VIEW (register.ejs)

- EJS file – Embedded JavaScript Template

- zaduženje: prezentacija dostavljenih podataka

MVC – CONTROLLER (router.js)

- povezuje Model I View

- prima zahtjev, pribavlja model, predaje ga u view kako bi se generirao HTML koji se onda vraća