Express.js

node.js

Zalety:

- asynchroniczność,
- szybkość działania – silnik JS V8 (np. w Google Chrome),
- wysoka skalowalność,
- npm,
- społeczność.

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Download for Windows (x64)

v6.10.2 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

v7.8.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Instalacja

express:

npm install -g express

express v4:

npm install express-generator –g

Tworzenie prostego serwera

npm init

inicjalizacja projektu poprzez stworzenie pliku **package.json**

```
"name": "simple-server",
"version": "1.0.0",
"description": "",
"main": "app.js",
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
},
"author": "",
"license": "ISC"
```

npm install express –save

instalacja express i dodanie do drzewa package.json

```
"author": "",
"license": "ISC",
"dependencies": {
  "express": "^4.15.2"
}
```

node app.js

Uruchamianie aplikacji app.js

```
var express = require('express');
var app = express();
var port = 3000;



app.listen(port, function(){
    console.log('server started');
});
```

npm install – g nodemon

Automatyczne wprowadzenie zmian, w razie zmiany zawartości plików serwera. konieczności ręcznego przeładowania przeładowywania.

nodemon app.js

Uruchomienie aplikacji w trybie nodemon.

```
server started
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
server started with nodemon
```

Dodanie prostej odpowiedzi http

req (request)

res (response)


```
var express = require('express');
var app = express();
var port = 3000;

app.get('/', function (req, res) {
    res.send('Hello World!')
});

app.listen(port, function(){
    console.log('server started');
});
```

Tworzenie projektu

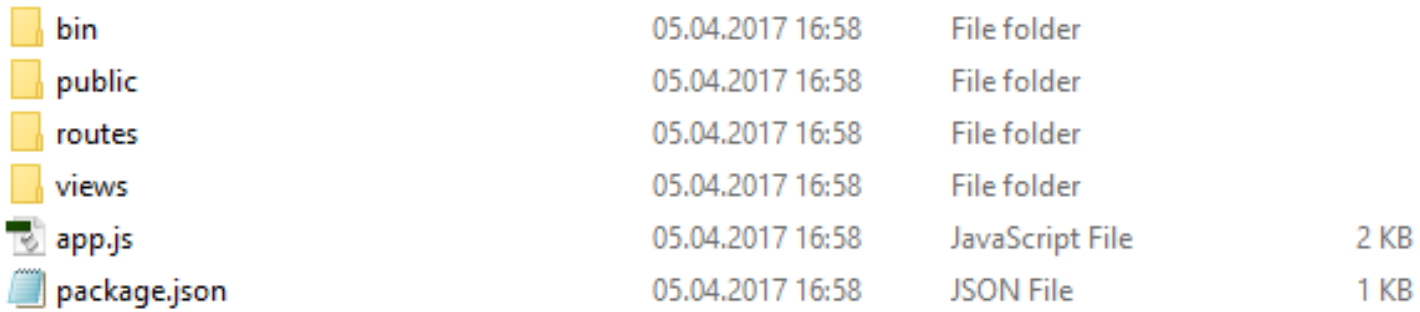
Inicjalizacja projektu poprzez komendę **express** w danym folderze.

```
express
```

```
create : v
create : v/package.json
create : v/app.js
create : v/public
create : v/routes
create : v/routes/index.js
create : v/routes/users.js
create : v/views
create : v/views/index.jade
create : v/views/layout.jade
create : v/views/error.jade
create : v/bin
create : v/bin/www
create : v/public/javascripts
create : v/public/images
create : v/public/stylesheets
create : v/public/stylesheets/style.css

install dependencies:
  $ cd v && npm install

run the app:
  $ DEBUG=v:* npm start
```



package.json

npm install

pobranie i instalacja wybranych modułów w folderze **node_modules**

```
{
  "name": "testapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.17.1",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.3",
    "express": "~4.15.2",
    "jade": "~1.11.0",
    "morgan": "~1.8.1",
    "serve-favicon": "~2.4.2"
  }
}
```

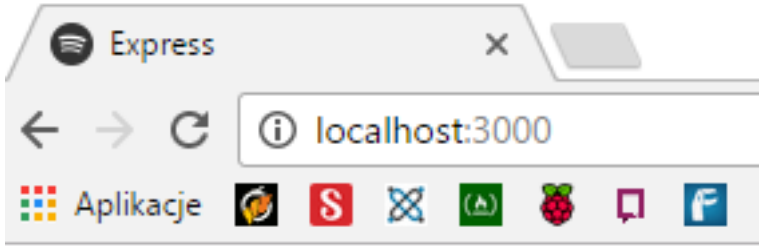
Uruchomienie serwera

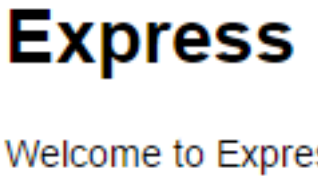
npm start – uruchomienie serwera

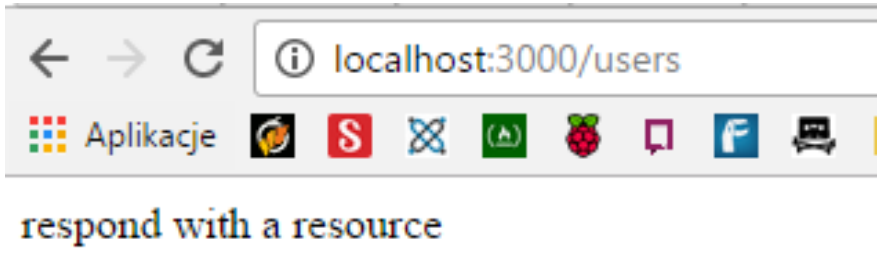
```
D:\programowanie\GTI\express\testApp>npm start

> testapp@0.0.0 start D:\programowanie\GTI\express\testApp
> node ./bin/www

GET / 200 926.009 ms - 170
GET / 200 27.569 ms - 170
GET /stylesheets/style.css 200 5.614 ms - 111
GET /favicon.ico 404 33.981 ms - 1175
GET / 304 26.006 ms - -
GET /stylesheets/style.css 304 1.590 ms - -
```







app.js

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var index = require('./routes/index');
var users = require('./routes/users');

var app = express();
```

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
```

```
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', index);
app.use('/users', users);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});
```

Routing

app.METHOD(PATH, HANDLER)

app – instancja express

method – metoda zapytania http(np. post, get, delete)

path – ścieżka na serwerze

handler – funkcja która jest wykonywana

Routing

```
app.get('/', function (req, res) {
  res.send('Hello World!')
});
```

```
app.post('/', function (req, res) {
  res.send('Got a POST request')
});
```

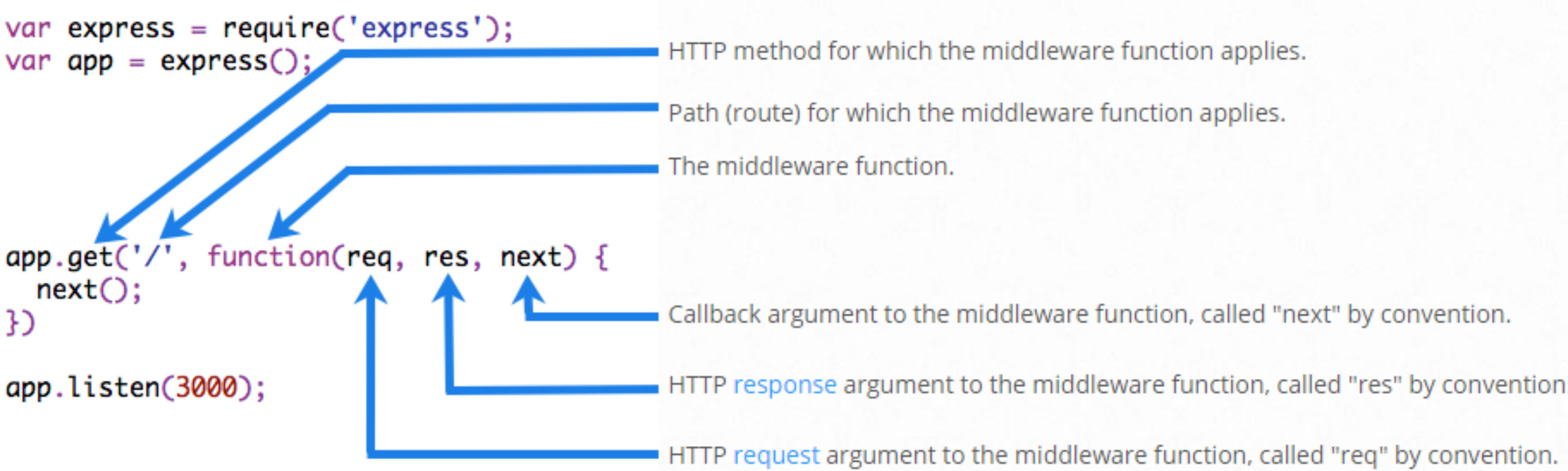
app.route()

```
app.route('/book')
    .get(function (req, res) {
      res.send('Get a random book')
    })
    .post(function (req, res) {
      res.send('Add a book')
    })
    .put(function (req, res) {
      res.send('Update the book')
    });
```

Metody Response

Method	Description
res.download()	Prompt a file to be downloaded.
res.end()	End the response process.
res.json()	Send a JSON response.
res.jsonp()	Send a JSON response with JSONP support.
res.redirect()	Redirect a request.
res.render()	Render a view template.
res.send()	Send a response of various types.
res.sendFile()	Send a file as an octet stream.
res.sendStatus()	Set the response status code and send its string representation as the response body.

Middleware


var express = require('express');
var app = express();
app.get('/', function(req, res, next) {
next();
})
app.listen(3000);
HTTP method for which the middleware function applies.
Path (route) for which the middleware function applies.
The middleware function.
Callback argument to the middleware function, called "next" by convention.
HTTP response argument to the middleware function, called "res" by convention.
HTTP request argument to the middleware function, called "req" by convention.

Middleware

```
var myLogger = function (req, res, next)
    console.log('LOGGED');
    next()
};

app.use(myLogger);

app.get('/', function (req, res) {
    res.send('Hello World!')
});
```

File: my-middleware.js

```
module.exports = function(options) {
  return function(req, res, next) {
    // Implement the middleware function based on the options object
    next()
  }
}
```

The middleware can now be used as shown below.

```
var mw = require('./my-middleware.js')

app.use(mw({ option1: '1', option2: '2' }))
```

MongoDB

1) Express nazwa_projektu
2) Package.json
   - Kerberos
   - Mongodb
3) npm install (npm install -g npm@latest)
4) mkdir data/db
5) Włączamy serwer **mongod**

MongoDB

6) use MongoDB

7) db.students.insert([{"student": "Zbyszek X", "ulica" : "brzozowa 2", "miasto": "poznan"}])

8)app.js

```
var mongo = require('mongodb');
```

MongoDB

9)

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/MongoDB', function (err, db) {
  if (err) throw err;

  db.collection('students').find().toArray(function (err, result) {
    if (err) throw err;

    console.log(result)
  })
});
```

MongoDB

```
D:\programowanie\GTI\express\MongoDB\mongoDB>npm start

> mongodb@0.0.0 start D:\programowanie\GTI\express\MongoDB\mongoDB
> node ./bin/www

[ { _id: 58e755fb1dce857c1b1ed717,
    student: 'Zbyszek X',
    ulica: 'brzozowa 2',
    miasto: 'poznan' } ]
```

Templates

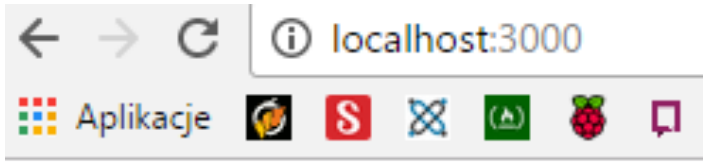
routes/index.js

```
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});
```

views/index.js

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```



Express

Welcome to Express