

# Express 4.0

Fast, unopinionated, minimalist web framework for Node.js



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Mi az Express?

- Node.js alapú open source webkiszolgáló keretrendszer
- Open source, egyik legnépszerűbb Node.js kiszolgáló
- Aszinkron HTTP/HTTPS kérések kezelés
- Alapvetően két szolgáltatása van
  - > routing
  - > middleware kezelés
- Számos kiegészítő pluginnel rendelkezik, nagy közösség áll mögötte
- Telepítés: `npm install express`

Express

Fast, unopinionated,  
minimalist web framework  
for Node.js

# Hello world!

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

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

var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;

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

# Express alkalmazás objektum

- Express alkalmazás, referencia az alábbi módon szerezhető meg:

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

- Ezen keresztül lehetséges:
  - > Konfiguráció
  - > Template kezelő motor regisztrálása
  - > Middleware-ek regisztrálása
  - > Routing objektumok regisztrálása

# HTTP kiszolgáló indítása

- Beépített szerver indítás támogatása

```
var express = require('express');  
var app = express();  
...  
app.listen(8080);
```

- Kézi szerver indítás

```
var express = require('express');  
var http = require('http');  
var app = express();  
var server = http.createServer(app);  
...  
server.listen(8080);
```

# Request objektum

- HTTP kérést reprezentáló objektum
- Tagváltozók:
  - > app: referencia az Express alkalmazásra
  - > route: referencia az aktuális Route objektumra
  - > baseUrl
  - > body: alapértelmezés *undefined*, body-parser tölti fel
  - > cookies: alapértelmezés *undefined*, cookie-parser tölti fel
  - > hostname, ip,
  - > params: url séma alapján mappelt paraméterek
  - > protocol
  - > query: Query string paraméterek
  - > ...

# Response objektum

- HTTP választ reprezentáló objektum
- Tagváltozók
  - > app: referencia az Express alkalmazásra
  - > locals: válaszhoz tartozó lokális változók, dinamikus nézetek használatához. Válaszba nem fog bekerülni!
- Függvények
  - > append: hozzáfűzés a válaszhoz
  - > cookie: cookie beállítások
  - > format: content-type szerinti válasz generálás
  - > send, render, json, redirect, end: Válasz elküldés, nézet renderelés, ...

# Routing alapok

- Routing meghatározza hogy az alkalmazás a különböző URI végpontokra hogyan reagáljon
- Egy útvonal (route) a következő elemekből épül fel:
  - > URI: elérési út
  - > HTTP method: GET, POST, PUT ...
- Az útvonal tartalmazhat paramétereket, komplex illesztési mintákat

```
app.get('/user/:id', function(req, res) {  
  res.send('user ' + req.params.id);  
});
```

- Middleware-ek definiálásával lehet megvalósítani
  - > Megadható egy adott útvonalra milyen middleware fusson le



# Statikus állományok kiszolgálása

- Cél: Képek, CSS, kliens oldali JavaScript, statikus HTML állományok kiszolgálása
- Megoldás: beépített `express.static` middleware

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

*//Példa kérések:*

```
http://localhost:3000/images/kitten.jpg
```

```
http://localhost:3000/css/style.css
```

```
http://localhost:3000/js/app.js
```

```
http://localhost:3000/images/bg.png
```

```
http://localhost:3000/hello.html
```

# Statikus állományok kiszolgálása

- Több mappa megadása

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

- Útvonal megadás:

```
app.use('/static', express.static(__dirname + '/public'));
```

*//Példa kérések:*

*http://localhost:3000/static/images/kitten.jpg*

*http://localhost:3000/static/css/style.css*

*http://localhost:3000/static/js/app.js*

*http://localhost:3000/static/images/bg.png*

*http://localhost:3000/static/hello.html*

# Middleware Express-ben

- A middleware egy függvény, ami hozzáfér:
  - > Request objektum
  - > Response objektum
  - > A request-response ciklus következő middleware-e (next objektum)
- Képességei:
  - > Bármilyen kód futtatása (aszinkron és szinkron is)
  - > Request és response objektumok változtatása
  - > Request-response ciklus megszakítása
  - > Következő middleware meghívása
    - Ha nem akarjuk megszakítani a ciklust akkor kötelező meghívni a next()-et

# Alkalmazás szintű middleware

- Közvetlenül az app objektumhoz köthetünk middleware-t

```
var app = express();

app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});
```

- Egy middleware a következő metódusokkal köthető a szülő objektumhoz
  - > use: minden kéreśnél lefut
  - > HTTP\_VERB: csak az adott típusú kéreśnél fut le  
VERB példák: get, post, put, patch, delete, options, head, ...

# Middleware-ek összefűzése

- A middleware-ek definiálásuk sorrendjében lesznek összefűzve
- Csoportosításuk a HTTP\_VERB és útvonal alapján történik

```
app.get('/user/:id', function (req, res, next) {  
  console.log('ID:', req.params.id);  
  next();  
}, function (req, res, next) {  
  res.send('User Info');  
});  
  
app.get('/user/:id', function (req, res, next) {  
  res.end(req.params.id);  
});
```

# Route szintű middleware

- Hasonlóan működik mint az alkalmazás szintű, csak a Router-ből származó objektumokon kell őket definiálni
- Segítségével egy alkalmazáson belül, több jól elszeparálható modult tudunk kialakítani

```
var app = express();  
var router = express.Router();  
  
router.use(function (req, res, next) {  
  console.log('Time:', Date.now());  
  next();  
});
```

# Hiba kezelő middleware

- Hasonlóan működik mint az alkalmazás és router szintű, azonban négy bementi paramétere van (és ezeket kötelező mint megadni), az alábbi sorrendben:
  - > error, request, response, next

```
app.use(function(err, req, res, next) {  
  console.error(err.stack);  
  res.status(500).send('Something broke!');  
});
```

# Gyakran használt middleware-ek

- body-parser
  - > HTTP kérések payload tartalmának feldolgozásához (JSON, Raw, Text, URL-Encoded)
- cookie-parser
  - > Cookie információk feldolgozása
- express-session, cookie-session
  - > Session kezelés
- passport
  - > Authentikációs modul
- serve-static, -index, -favicon
  - > Statikus állományok kiszolgálására
- compress
  - > Kérések gzip/deflate tömörítése



# Adatbázis kapcsolat

- Tetszőleges Node.js adatbázis driver-t vagy ORM-et lehet használni
  - > Inicializációkat az alkalmazás belépési pontján (tipikusan server.js) lehet elvégezni

```
var db = require('mongoskin')
        .db('mongodb://localhost:27017/animals');

db.collection('mamals').find().toArray(function(err, result) {
  if (err) throw err;
  console.log(result);
});
```

# Express gyakorlat