

Wybrane elementy praktyki projektowania oprogramowania
Wykład 11/15
node.js: Express (4)

Wiktor Zychła 2018/2019

1	Spis treści	
2	AJAX.....	2
3	WebSockets.....	4
4	Deployment.....	7

2 AJAX

AJAX = Asynchronous Javascript And XML – ogólna nazwa techniki w której przeglądarka robi dodatkowe żądania do serwera nie przez nawigację (GET) lub wysyłanie formularzy (POST) ale za pomocą Javascript i obiektu [XmlHttpRequest](#).

Akronim wywodzi się z tego że w oryginalnej implementacji z serwera pobierano dane w postaci XML. W praktyce jest to niewygodne, ponieważ odczyt XML nie ma wsparcia w Javascript po stronie klienta (przeglądarki).

W praktyce częściej **AHAH** lub **AJAJ**

AHAH = Asynchronous HTML and HTTP – serwer zwraca HTML który jest dynamicznie dodawany w odpowiednie miejsce drzewa DOM za pomocą Javascript w przeglądarce

AJAJ = Asynchronous Javascript and JSON – serwer zwraca dane w postaci JSON, które są dynamicznie przetwarzane i zamieniane na elementy drzewa DOM za pomocą Javascript w przeglądarce

```
// app.js
var http = require('http');
var express = require('express');
var ejs = require('ejs');
var multer = require('multer');

var app = express();
var upload = multer();

app.set('views', './views');
app.set('view engine', 'html');

app.engine('html', ejs.renderFile );

app.post('/ajax', upload.single(), (req, res) => {

    var txtParam = req.body.txtParam;
    res.end(`<div>zawartość z serwera ${txtParam}</div>`);
});

app.get('/', (req, res) => {

    res.render('app', { message : 'dynamiczne dane 2' } );
});

http.createServer(app).listen(process.env.PORT || 3000);
```

```
<!-- views/app.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    window.addEventListener('load', function() {
      var bt = document.getElementById('btApp');
      bt.addEventListener('click', function() {
        var content = document.getElementById('content');
        var param = document.getElementById('txtParam');
        //content.innerHTML = 'nowa zawartość modyfikowana';

        var req = new XMLHttpRequest();
        req.open('post', '/ajax', true);
        req.onreadystatechange = function()
        {
          if ( req.readyState == XMLHttpRequest.DONE )
          {
            content.innerHTML = req.responseText;
          }
        }
        var form = new FormData();
        form.append('txtParam', param.value);
        req.send(form);
      });
    });
  </script>
</head>
<body>
  kawałek statyczny
  <div id='content'>
    stara zawartość
  </div>
  <input type='text' id='txtParam'> <br/>
  <button id='btApp'>odśwież zawartość</button>
</body>
</html>
```

3 WebSockets

WebSockets – duplexowy (dwukierunkowy) protokół komunikacyjny, oparty o TCP. Wbudowany we współczesne przeglądarki.

Protokół WebSockets inicjuje się z przeglądarki żądaniem z dodatkowym nagłówkiem

```
Upgrade: websocket
```

na który serwer reaguje odesłaniem HTTP 101 (Switching protocols) i ustanowieniem osobnego gniazda komunikacyjnego.

Tworzenie aplikacji opartych o WebSockets możliwe jest w [podstawowym Javascript w przeglądarce](#), aczkolwiek jest dość uciążliwe, stąd szereg „opakowań” ułatwiających pisanie kodu klienta i serwera. Dla node.js najpopularniejszą biblioteką dla WebSockets jest [socket.io](#).

Socket.io zrewolucjonizował swego czasu tworzenie interaktywnych aplikacji internetowych z uwagi na ogromną łatwość modelu programowania oraz obsługę tzw. [fallback](#) czyli sytuacji w której przeglądarka nie posiada wsparcia dla WebSockets.

```
// app.js
var http = require('http');
var socket = require('socket.io');
var fs = require('fs');
var express = require('express');

var html = fs.readFileSync('app.html', 'utf-8');

var app = express();
var server = http.createServer(app);
var io = socket(server);

app.use( express.static('./static'));

app.get('/', function(req, res) {
  res.setHeader('Content-type', 'text/html');
  res.write(html);
  res.end();
});

server.listen(process.env.PORT || 3000);

io.on('connection', function(socket) {
```

```

console.log('client connected:' + socket.id);
socket.on('chat message', function(data) {
    io.emit('chat message', data); // do wszystkich
    //socket.emit('chat message', data); tylko do połączanego
})
});

setInterval( function() {
    var date = new Date().toString();
    io.emit( 'message', date.toString() );
}, 1000 );

console.log( 'server listens' );

```

```

<!-- app.html -->
<html>
  <meta charset="utf-8" />
  <head>
    <script src="/socket.io/socket.io.js"></script>
    <script src="/appjs.js"></script>
    <script>
      window.addEventListener('load', function() {

        var socket = io();
        socket.on('message', function(data) {
          var t = document.getElementById('time');
          t.innerHTML = data;
        });

        socket.on('chat message', function(data) {
          var msg = document.getElementById('messages');
          msg.innerHTML += data + "<br/>";
        });

        var btsend = document.getElementById('btsend');
        btsend.addEventListener('click', function() {
          var txtmessage = document.getElementById('txtmessage');
          socket.emit('chat message', txtmessage.value);
        });

      });
    </script>
  </head>
  <body>
    Aktualny czas na serwerze :) <b><span id="time"></span></b>.

    <div>
      <input type='text' id='txtmessage' />

```

```
<button id='btsend'>Wyślij</button>  
<div id='messages'></div>  
</div>  
</body>  
</html>
```

4 Deployment

Deployment = umieszczenie aplikacji w środowisku serwerowym z którego jest hostowana w sieci

Continuous Deployment = technika organizacji sposobu publikacji aplikacji, w którym dostarczenie aplikacji do środowiska z którego jest hostowana jest zautomatyzowane (i być może wplecione w proces kompilacji)

Z punktu widzenia aplikacji node.js, łatwość Continuous Deployment polega na tym, że Javascript nie wymaga kompilacji. W efekcie, w przeciwieństwie do wielu innych języków / technologii, na serwer trafia kod źródłowy.

Umożliwia to ciekawe podejście w którym zasób sieciowy repozytorium kodu (np. GIT) jest jednocześnie miejscem z którego aplikacja jest udostępniana.

Hosting node.js świadczą obecnie wszyscy duzi i wielu małych dostawców:

- Microsoft w chmurze [Azure](#)
- Google w chmurze [App Engine](#)
- Amazon w [AWS](#)

My poznamy możliwość darmowego hostingu node.js na [Heroku](#).

The image shows a screenshot of the Heroku website. The browser's address bar displays 'Heroku, Inc. (US) https://www.heroku.com'. The website header includes the Heroku logo and navigation links: 'Products', 'Marketplace', 'Pricing', 'Documentation', 'Support', and 'More'. There is also a search bar and 'Log in or Sign up' options. The main content area features a diagram of a deployment pipeline with stages 'REVIEW', 'STAGING', and 'PRODUCTION', and a laptop displaying code. Text on the page reads: 'DEVELOPERS Focus on your apps Invest in apps, not ops. Heroku handles the hard stuff — patching and upgrading, 24/7 ops and security, build systems, failovers, and more — so your developers can stay focused on building great apps.' Below this is a 'SIGN UP FOR FREE' button and a link to 'Explore the Heroku Platform'. At the bottom, there are logos for various programming languages (JS, Ruby, Python, PHP, etc.) and a 'Heroku Podcasts' section with a 'LISTEN NOW' button.

Do zahostowania aplikacji na Heroku niezbędne są:

- [Klient Git](#)
- [Heroku CLI](#)

Instrukcja

1. W package.json dodać scripts/start

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "start": "node app.js"  
}
```

2. Utworzyć plik tekstowy `.gitignore` z wpisanym `node_modules` (folder ignorowany przy commit/push)

```
node_modules
```

3. Zmodyfikować kod dodając możliwość przekazania numeru portu do nasłuchu lokalnego (serwer Heroku przekazuje numer portu w zmiennej środowiskowej):

```
server.listen(process.env.PORT || 3000);
```

4. Z linii poleceń wykonywać skrypt (linia po linii)

```
git init  
git add .  
git commit -m "initial commit"  
heroku login  
heroku create  
git push heroku master
```

Po wykonaniu poszczególnych kroków, aplikacja działa na Heroku. [Pełna dokumentacja integracji z node.js.](#)