OPANUJ EXPRESS.JS I WYKORZYSTAJ GO W PROGRAMOWANIU

Apress

Express.js

Tworzenie aplikacji sieciowych w Node.js

Azat Mardan

Helion

Tytuł oryginału: Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-1664-5

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: **http://helion.pl** (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/expres
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 19

Galeria Instagram

Jeżeli kolejne rozdziały książki czytasz po kolei, to poznałeś już ważne, choć suche szczegóły API oraz miałeś styczność z jedynie abstrakcyjnymi rozwiązaniami. Część IV powinna okazać się najbardziej ekscytująca, ponieważ cztery znajdujące się tutaj rozdziały zostały poświęcone programowaniu i przykładom.

W tym rozdziale dowiesz się, jak używać Express.js wraz z usługami zewnętrznymi opracowanymi przez firmy trzecie (tutaj to API Storify). Celem budowanej aplikacji jest pobranie zdjęć Instagram ze Storify, a następnie wyświetlenie ich w galerii. Poza frameworkiem Express.js wykorzystamy jeszcze następujące moduły:

- `superagent` (*https://www.npmjs.com/package/superagent*);
- `consolidate` (*https://www.npmjs.com/package/consolidate*);
- `handlebars` (*https://www.npmjs.com/package/handlebars*).

Zdecydowałem się na wymienione moduły, ponieważ są one popularne w pewnych kręgach programistycznych Node.js. Istnieje więc duże prawdopodobieństwo, że je napotkasz lub będziesz z nich korzystał w przyszłości.

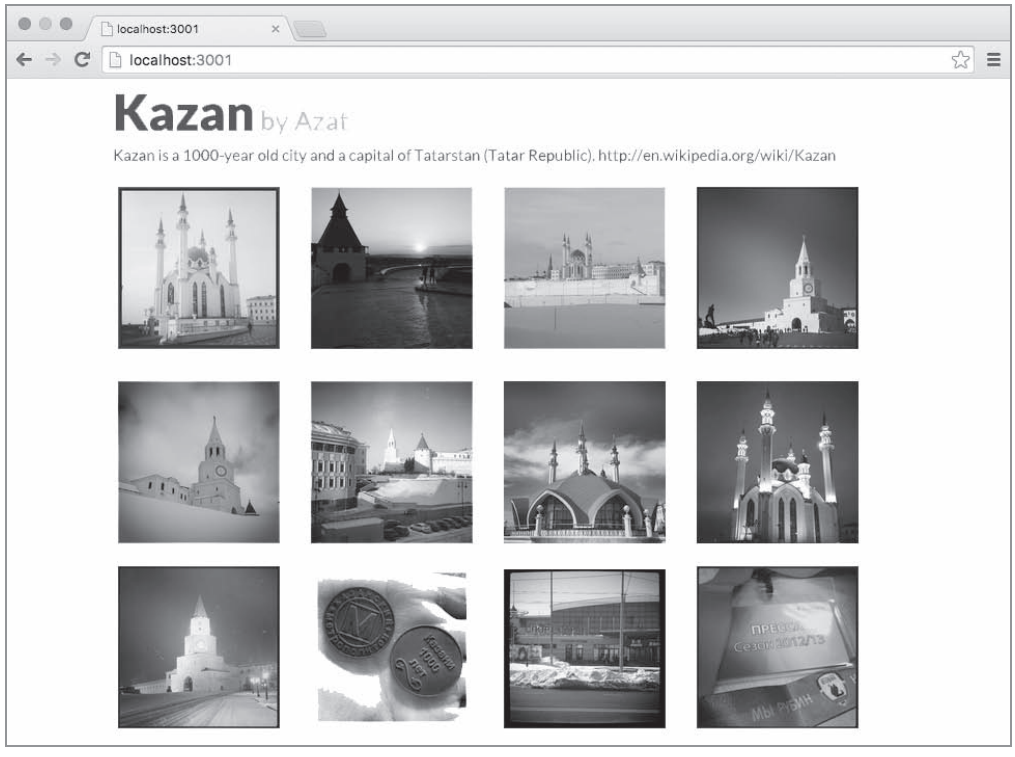
Uwaga Pełny kod źródłowy przykładu omawianego w rozdziale znajdziesz pod adresem: *ftp://ftp.helion.pl/przyklady/expres.zip*.

Storify (*http://storify.com/*) działa w Node.js (*https://nodejs.org/*) i Express.js (*http://expressjs.com/*). Dlatego też wymienione technologie można wykorzystać do utworzenia aplikacji pokazującej, jak budować rozwiązania opierające się na żądaniach HTTP i API firm trzecich.

Zaczynamy pracę nad galerią Instagram

Galeria Instagram będzie pobierać obiekt, a następnie wyświetlać jego tytuł, opis i elementy/obrazy, podobnie jak pokazałem na rysunku 19.1.

Uwaga Jeżeli zastanawiasz się, co oznacza słowo Kazan (Kazań), wyjaśniam, że to ponadtysiącletnia stolica Republiki Tatarstanu.



Rysunek 19.1. *Galeria Instagram*

Struktura plików aplikacji przedstawia się następująco:

```
- index.js
- package.json
- views/index.html
- css/bootstrap-responsive.min.css
- css/flatly-bootstrap.min.css
```

Pliki CSS pochodzą z biblioteki Bootstrap (*http://getbootstrap.com/*) i motywu Flatly (*http://bootswatch.com/flatly/*). Z kolei *index.js* to nasz główny plik Node.js zawierający większość logiki, natomiast *index.html* to szablon Handlebars. Aplikacja używa zwykłych arkuszy stylów CSS zdefiniowanych w dwóch plikach znajdujących się w katalogu *css*.

Poniżej wymieniłem zależności aplikacji:

- `express` w wersji 4.8.1 — framework Express.js;
- `superagent` w wersji 0.18.2 — do wykonywania żądań HTTP(S);
- `consolidate` w wersji 0.10.0 — w celu użycia silnika szablonów Handlebars w Express.js;
- `handlebars` w wersji 2.0.0-beta.1 — w celu użycia silnika szablonów Handlebars.

Zawartość pliku *package.json* przedstawia się następująco:

```
{
  "name": "sfy-gallery",
  "version": "0.2.0",
  "description": "Galeria Instagram: oparty na Node.js przykład użycia API Storify",
  "main": "index.js",
```

```
  "scripts": {
    "test": "echo \"Błąd: nie podano testu.\" && exit 1"
  },
  "dependencies": {
    "consolidate": "0.10.0",
    "express": "4.8.1",
    "handlebars": "2.0.0-beta.1",
    "superagent": "0.18.2"
  },
  "repository": "https://github.com/storify/sfy-gallery",
  "author": "Azat Mardan",
  "license": "BSD"
}
```

Aby zainstalować niezbędne moduły, wydaj poniższe polecenie:

```
$ npm install
```

Teraz utwórz plik *index.js*. Na początku wymienionego pliku powinny znaleźć się następujące zależności:

```
var express = require('express');
var superagent = require('superagent');
var consolidate = require('consolidate');

var app = express();
```

Kolejnym krokiem jest konfiguracja silnika szablonów:

```
app.engine('html', consolidate.handlebars);
app.set('view engine', 'html');
app.set('views', __dirname + '/views');
```

Dalej przechodzimy do przygotowania katalogu statycznego wraz z oprogramowaniem pośredniczącym:

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

Jeżeli chcesz wykorzystać inną galerię, możesz to zrobić. Potrzebujesz jedynie nazwy użytkownika autora oraz tzw. *story slug*. W przypadku mojej galerii o Kazaniu (stolicy Tatarstanu) podaj następujące dane:

```
var user = 'azat_co';
var story_slug = 'kazan';
```

Następnie umieść swoje wartości: klucz API Storify, nazwę użytkownika i token, jeśli go masz. W chwili pisania książki API Storify pozostało publiczne, co oznacza *brak konieczności przeprowadzania uwierzytelniania* (nie trzeba używać klucza). Jeżeli w przyszłości sytuacja ulegnie zmianie, żądanie klucza API można będzie złożyć na stronie: *http://dev.storify.com/request*. Zawsze też możesz zajrzeć do oficjalnej dokumentacji dostępnej na stronie: *http://dev.storify.com/api/summary*.

```
var api_key = "";
var username = "";
var _token = "";
```

Kolejnym krokiem jest zdefiniowanie trasy głównej (`/`):

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

Elementy z API Storify pobieramy w wywołaniu zwrotnym trasy za pomocą metody `superagent.get()`:

```
superagent.get("http://api.storify.com/v1/stories/"
  + user + "/" + story_slug)
```

Punktem końcowym API Storify jest `"http://api.storify.com/v1/stories/" + user + "/" + story_slug`, czyli w omawianym przykładzie to: *https://api.storify.com/v1/stories/azat/kazan*. Jedną z zalet metody `superagent` jest możliwość łączenia metod. Na przykład metoda `query()` powoduje wysłanie danych w ciągu tekstowym zapytania:

```
.query({api_key: api_key,
  username: username,
  _token: _token})
```

Metoda `set()` pozwala na określenie nagłówków żądania:

```
.set({Accept: 'application/json'})
```

Z kolei metoda `end()` pobiera wywołanie zwrotne przeznaczone do wykonania po otrzymaniu odpowiedzi:

```
.end(function(e, storifyResponse){
  if (e) return next(e);
```

Aby wygenerować szablon z obiektem wskazywanym przez właściwość `content` odpowiedzi HTTP, możemy użyć poniższego fragmentu kodu:

```
      return res.render('index', storifyResponse.body.content);
    })
})

app.listen(3001);
```

API Storify zwraca dane w formacie JSON. Informacje o używanym formacie znajdziesz na stronie: *https://api.storify.com/v1/stories/azat_co/kazan* (przyjmując założenie, że API nadal będzie publiczne, jak w chwili pisania książki). Zwięzłą (tzn. niepokazującą wszystkich zagnieżdżonych obiektów) postać danych JSON pokazałem na rysunku 19.2.

Rysunek 19.2. *Przykład zwięzłych danych wyjściowych dostarczanych przez API Storify dla encji albumu*

Wyświetlanie galerii

Skoro przygotowaliśmy aplikację pobierającą dane ze Storify i wywołującą szablon `index` w celu ich wyświetlenia, warto spojrzeć na szablon Handlebars, który znajduje się w pliku *views/index.html*:

```
<!DOCTYPE html lang="en">
<html>
  <head>
    <link type="text/css"
      href="css/flatly-bootstrap.min.css"
      rel="stylesheet" />
    <link type="text/css"
      href="css/bootstrap-responsive.min.css"
      rel="stylesheet"/>
  </head>

  <body class="container">
    <div class="row">
```

Teraz użyjemy `{{title}}` do wyświetlenia tytułu albumu Storify oraz `{{authon.name}}` do wyświetlenia autora:

```
<h1>{{title}}<small> by {{author.name}}</small></h1>
  <p>{{description}}</p>
</div>
<div class="row">
  <ul class="thumbnails">
```

Kolejnym krokiem jest wykorzystanie wbudowanej w Handlebars konstrukcji do przeprowadzenia iteracji przez elementy tablicy. W trakcie każdej iteracji następuje wygenerowanie nowego znacznika `<li>`:

```
      {{#each elements}}
        <li class="span3">
          <a class="thumbnail" href="{{permalink}}"
          target="_blank">
            <img src="{{data.image.src}}"
              title="{{data.image.caption}}" />
          </a>
        </li>
      {{/each}}
      </ul>
    </div>
  </body>

</html>
```

Po uruchomieniu aplikacji za pomocą polecenia `node .` i przejściu pod adres: *http://localhost:3000* zobaczysz wyświetlone zdjęcia. Działanie aplikacji jest następujące: po przejściu na podaną stronę serwer lokalny wykonuje żądania do Storify i pobiera z galerii Instagram łącza do zdjęć.

Podsumowanie

Framework Express.js i moduł `superagent` pozwalają programistom na pobieranie danych dostarczanych przez usługi opracowane przez firmy trzecie, takie jak: Storify, Twitter i Facebook, za pomocą jedynie kilku wierszy kodu. Umożliwiają też zarządzanie tymi danymi. Przykład przedstawiony w rozdziale jest prosty, ponieważ nie wykorzystuje bazy danych. W kolejnym rozdziale przystąpimy do budowy aplikacji Todo, która zostanie oparta na bazie danych MongoDB.

■ **Uwaga** W większości przypadków dostawcy usług (np.: Google, Facebook i Twitter) wymagają uwierzytelnienia (w chwili pisania książki nie było takiego wymogu dla API Storify). Aby wykonywać żądania: OAuth 1.0, OAuth 2.0 i OAuth Echo, rozważ użycie modułów: `oauth` (*https://www.npmjs.com/package/oauth*; GitHub: *https://github.com/ciaranj/node-oauth*), `everyauth` (*https://www.npmjs.com/package/everyauth*; GitHub: *https://github.com/bnoguchi/everyauth*) i/lub `passport` (*http://passportjs.org/*; GitHub: *https://github.com/jaredhanson/passport*).

Skorowidz

K

L

M

N

O

P

R

S

Ś

T

U

W

Z

Ż

WYPRÓBUJ EXPRESS.JS — PRACUJ WYGODNIE, WYDAJNIE I EKSPRESOWO!

Express.js

Tworzenie aplikacji sieciowych w Node.js

Frameworku Express.js używa wiele znanych firm, takich jak MySpace i Storify, które dostrzegły ogromne korzyści płynące z wykorzystywania tej technologii, a także doceniły jej stabilność i bezpieczeństwo. Przed Express.js i Node.js otwiera się świetlana przyszłość. Oznacza to, że jako ekspert w zakresie Node.js, biegle posługujący się Express.js, staniesz się poszukiwanym specjalistą!

Ta książka jest kompleksowym podręcznikiem, który przedstawi Ci sposób działania Express.js w praktyce i przeprowadzi Cię przez poszczególne etapy budowy aplikacji. Autor jasno i precyzyjnie wyjaśnia wszystkie koncepcje, których zrozumienie jest niezbędne do programowania w Express.js. Znajdziesz tu omówienie zagadnień związanych m.in. z oprogramowaniem pośredniczącym, tworzeniem szkieletu aplikacji, generowaniem szablonów, przetwarzaniem danych żądania i cookies, zarządzaniem uwierzytelnianiem i sesjami, obsługą błędów i przygotowaniem aplikacji do wdrożenia w środowisku produkcyjnym. Na pewno docenisz też liczne przykłady kodu źródłowego.

Jeśli — jako inżynier oprogramowania lub programista sieciowy — poszukujesz sposobu, aby bez wertowania setek stron dokumentacji programować z wykorzystaniem frameworku Express.js, a przy tym poszerzyć swoje kompetencje, ta książka jest właśnie dla Ciebie.

Dzięki tej książce:

- rozpoczniesz pracę z oprogramowaniem pośredniczącym
- będziesz sprawnie korzystać z silników szablonów
- dowiesz się, jak zapewnić aplikacji bezpieczeństwo
- nauczysz się tworzyć aplikacje oparte na domenie
- przeanalizujesz przykłady działających aplikacji, takich jak Instagram, HackHall i inne

Azat Mardan — od kilkunastu lat programuje aplikacje mobilne i sieciowe. Pracował nad wieloma aplikacjami o kluczowym znaczeniu, wykorzystywanymi przez agencje rządowe USA. Jest twórcą kilku projektów open source dla Node.js (m.in. ExpressWorks, mongoui, HackHall.com i NodeFramework.com), a także współtwórcą express, oauth, jade-browser i innych modułów npm. Tworzy programistyczne kursy online, publikuje specjalistyczne artykuły na blogu, a ponadto jest autorem książek poświęconych JavaScriptowi i Node.js.

Helion
38252 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Apress

ISBN 978-83-283-1664-5
9 788328 316645
cena: 59,00 zł