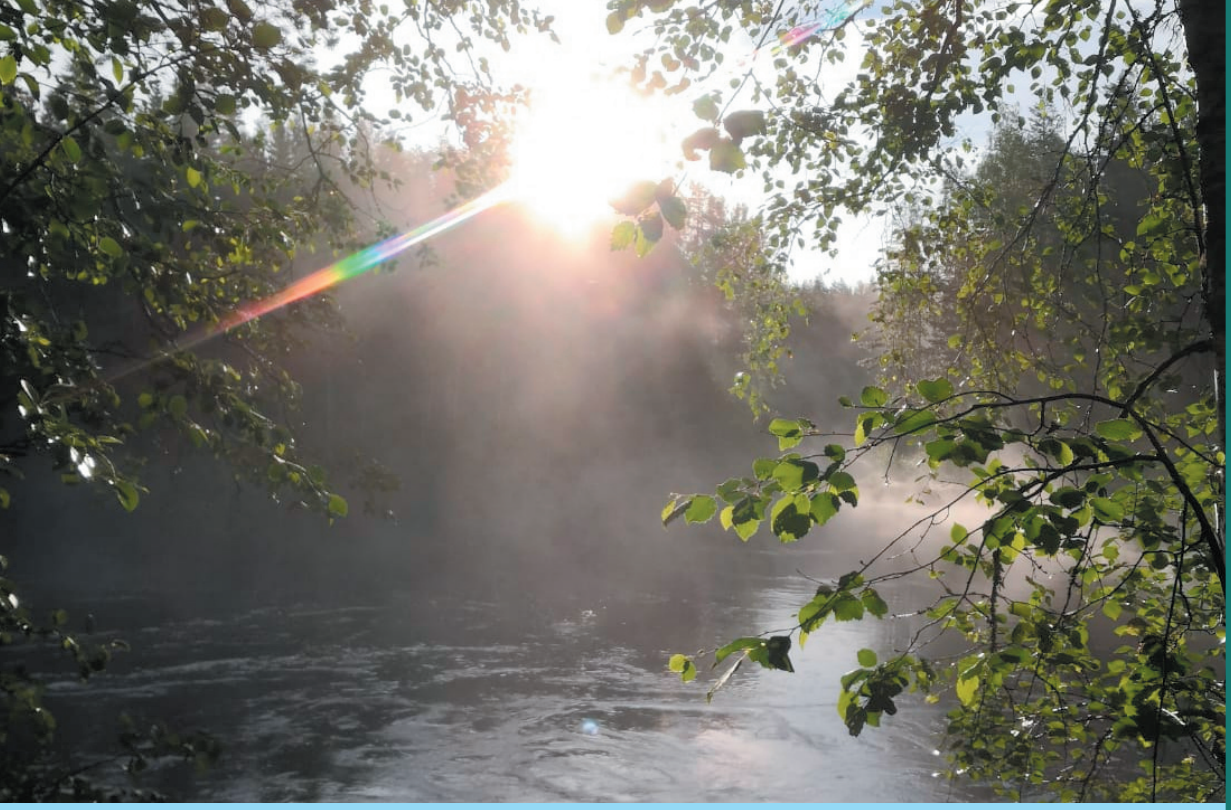

Juha Peltomäki

Node.js

Web-palveluiden ohjelmointi

Web-palveluiden ohjelmointi JavaScriptillä

Node.js

Web-palveluiden ohjelmointi

Juha Peltomäki

Kustantaja: BoD – Books on Demand, Helsinki, Suomi

Valmistaja: BoD – Books on Demand, Norderstedt, Saksa

ISBN: 978-952-802-634-1

Sisällysluettelo

Node.js

Node on suosittu vaihtoehto sovellusten tekemiseen, jossa tarvitaan paljon rinnakkaisuutta mutta vähemmän prosessointia palvelinpuolella. Interaktiiviset ja reaaliaikaiset verkkosovellukset, joihin tulee paljon I/O-pyyntöjä, ovat otollinen kohde Noden käyttämiseen. Jos vastaavasti tarvitaan raskasta prosessointia, joka suoritus kestää pitkään, Node ei välttämättä ole paras vaihtoehto, koska se on yksisäikeinen.

Kaiken kaikkiaan Noden keveys voi merkittävästi parantaa verkkosovelluksen skaalautuvuutta, ylläpidettävyyttä ja tehokkuutta. Nodea voi käyttää modularisoimaan yrityksen sovelluspino sekä jakamaan sovelluskehitysprosessi lukuisiksi, toisistaan erillisiksi mikropalveluiksi. Lisäksi tällaista sovellusta voi aiempaa skaalautuvammin ajaa pilviympäristössä.

Node ja MEAN

NODE on **MEAN**-sovelluspinossa merkki "N" ja toimii **MEAN Stack** -pinolla kehitettyjen sovellusten ajoympäristönä (*runtime environment*). Itse MEAN-pinossa kaikki on ohjelmoitu JavaScriptillä, mikä tekee sovelluskehityksestä yksinkertaisen ja helpon sekä kehittäjän että ohjelmien ajon kannalta. MEAN-pinoon voi integroida myös useita muita moduuleja. Esimerkiksi Mongoose-kirjastoa voi käyttää Noden rinnalla helpottamaan integrointia MongoDB:n kanssa.

Keskeiset käyttökohteet

Node.js (usein kutsutaan pelkästään **Node**) tärkeimpiä käyttökohteita ovat:

- Streaming-tietovirtaa hyödyntävät sovellukset
- tapahtumapohjaiset reaaliaikasovellukset
- Yhden sivun sovellukset Single-Page Application (SPA)

- reaaliaikaiset chat-sovellukset

stream-tietovirtatuki

Node sisältää sisäänrakennettuna rajapinnan sekä kirjoitettaville ja luettaville **tietovirroille**. Niitä voidaan myös helposti ja tehokkaasti prosessoida ja monitoroida.

Node sisältää natiivisti Stream-moduulin, joka toimitetaan core noden mukana. Nodessa **Stream** on **EventEmitter**-luokan toteutus, ja sillä käsitellään tapahtumat asynkronisesti Node-sovelluksessa. Tästä syystä tietovirrat ovat Nodessa tapahtumapohjaisia (*event-based streams*).

Reaaliaikasovellukset

Reaaliaikaiset sovellukset (*real-time applications*) ovat pohjimmiltaan ohjelmistoratkaisuja, joita käytetään laajasti esimerkiksi projektinhallintaan, ääni- ja videoneuvotteluihin, yhteistyössä tehtäviin asiakirjojen ja dokumenttien muokkaamiseen, keskustelusovelluksiin, piirtosovelluksiin ja muihin vastaaviin sovelluksiin.

Reaaliaikaisissa ohjelmistoissa tapahtuu samanaikaisesti paljon I/O-pyyntöjä ja tapahtumia. Esimerkiksi projektinhallintaratkaisussa useat käyttäjät voi kirjoittaa kommentteja, tai he voivat päivittää projektin tilaa ja liittää kuvia, videoita ja muuta mediaa tiettyyn projektiin. Tämän tyyppisiin sovelluksiin **tapahtumapohjainen** ja **stream-tietovirtoja** hyvin tehokkaasti tukeva Node on erittäin nopea ja toimiva ratkaisu.

Node käsitellään I/O-operaatioita tehokkaasti käyttäen mm. sisäänrakennettuja **WebSocket API**- sekä **Event API**-rajapintojaan. Tämän takia Nodesta on tullut sovelluskehittäjien suosikkivalinta reaaliaikaisten ja tapahtumapohjaisten yhteistyösovellusten rakentamiseen.

Yhden sivun sovellukset

Node sopii erinomaisesti yksisivuisten **SPA-sovellusten** (*Single Page Application*) tekemiseen, koska se pystyy käsittelemään tehokkaasti asynkronisia pyyntöjä ja

ajamaan rinnakkain raskaita I/O-toimintoja. Node toimii hyvin dataohjattujen yksisivuisten SPA-sovellusten kanssa, joissa palvelin toimii taustalla siirtäen tiedon asiakkaana toimivalle web-selaimelle, joka vastaavasti hoitaa kaiken HTML-renderöinnin.

SPA-sovellusten rakentamisen suurin etu monille kehittäjille on se, että kaikki toteutukset voi hoitaa yhdellä kielellä, JavaScriptillä. Tällöin ainakin osittain samat ratkaisut (luokat, tietorakenteet, funktiot jne.) ja jopa toteutukset voivat olla käytössä sekä asiakas- että palvelinpuolella. Yhden kielen käyttäminen nopeuttaa sekä sovelluksen kehittämistä että ylläpitoa.

Microservices-ratkaisut

Noden avulla voi kehittää web-palvelinsovelluksia ja usein sovellus rakennetaan palveluarkkitehtuurin pohjaisesti. Tällöin puhutaan **RESTful**-palveluista tai usein myös **MicroServices**-arkkitehtuurista, joka toki pohjautuu RESTful-palveluin.

Microservices-arkkitehtuuri on ratkaisu, jossa web-sovellus kehitetään pieninä, riippumattomina palveluina, joissa jokainen palvelu toteuttaa jonkin yksittäisen, toisistaan riippumattoman osan liiketoimintalogiikasta.

Microservices-arkkitehtuuri on tullut yhä suositummaksi ja sitä käyttää monissa palveluissa mm. Amazon, Facebook, Twitter ja Netflix.

Tämän tyyppisissä web-sovelluksissa viestitään tavallisesti käyttäen REST/HTTP-protokollaa sekä JSON-tietorakenteita.

Node web-palvelinsovelluksen kehittämiseen

Nodea voi käyttää palvelinpuolen toteuttamisen. Palvelinpuolelle JavaScript-ohjelmia voi rakentaa käyttäen Node-ympäristöä. Se sisältää JavaScript-kielen tuen, joka on rakennettu **Chromium**-projektin ja **Google Chromen** käyttämän **V8**-selainmoottorin varaan. npm-paketinhallintajärjestelmällä voi ylläpitää ja hallita uusien ominaisuuksien ja pakettien asentamista node-ympäristöön ja pitää näin ollen järjestelmän kirjastot ja niiden versiot ajan tasalla.

Ryan Dahlin vuonna 2009 Linuxille julkaisema Node.js -versio mahdollisti JavaScript-kieltä käyttäen tapahtumapohjaisen ohjelmoinnin palvelinpuolella. Julkaisun merkityksellisyydestä kertoo se, että vuonna 2011 Microsoft julkaisi Node.js -version myös Windows-käyttöjärjestelmään.

Nodea on suosittu ratkaisu erityyppisten ja usein palvelupohjaisten web-sovellusten rakentamiseen. Keskeisiä huomioitavia asioita web-sovellusten rakentamisessa ovat:

- **Core Node.js** sellaisenaan riittämätön ratkaisu web-palvelinsovellusten tai web-palveluiden rakentamiseen. Core Node sisältää pääasiassa vain JavaScript-ohjelmien suorittamisen asynkronisesti palvelinpuolella.
- Todellisten web-sovellusten kehittäminen ja suorittaminen vaatii yleensä jonkin Node Frameworkin eli Node-sovelluskehyksen
- Esimerkiksi web-sovellusten tekemiseen oiva ratkaisu on jonkin **Web Framework**, kuten esimerkiksi tässä kirjassa käytettävä **Express JS**: http://expressjs.com/

MEAN-sovelluspino

Usein Noden yhteydessä puhutaan **MEAN**-sovelluspinosta, joka on kokonaisuus Full-Stack web-sovellusten rakentamiseen. Se sisältää seuraavat sovellukset:

- **MongoDB**,
- **ExpressJS**,
- **AngularJS** ja
- **Node**.

MEAN-sovelluspinossa on keskeisenä komponenttina yhteinen ohjelmointikieli eli JavaScript. Sekä asiakaspuoli (*frontend*) että palvelinpuoli (*backend*) kehitetään JavaScriptillä, jolloin toteutukset voi kytkeä paljon tiiviimmin toisiinsa. Tiedonsiirtoon käytetään yleensä JSON-formaattia, joka on natiivia JavaScriptiä, joten sitä on helppo ja nopea jäsentää sekä palvelin- että asiakaspäässä käyttäen osittain samoja luokkia.

Node.js on tärkein ja yleisin palvelinpuolen JavaScript-sovelluskehys. Node toi JavaScriptiin mm. tuen Web-palveluiden toteuttamiseen. Noden avulla koko sovellus voidaan toteuttaa alusta loppuun JavaScriptin avulla. Node.js tarjoaa pohjan web-sovellusten kehittämiseen, mutta Express Framework tarjoaa sovelluskehyksen Web-sovelluskehitykseen. Noden avulla kokonaiset Full Stack Web-sovellukset voidaan rakentaa pelkästään JavaScript-kieltä käyttäen.

MongoDB sopii ei-rakenteisen datan tallentamiseen ja sitä käytetään usein Node-sovellusten tallennusratkaisuna. Kirjassa käydään läpi Nodella toteutettuja REST-palveluita, jotka tallentavat tietonsa MongoDB-tietokantaan.

Juha Peltomäki on kokenut ohjelmoinnin kouluttaja, sovelluskehittäjä sekä tietokirjailija. Hän on kirjoittanut yli kymmenen ohjelmointikirjaa. Hänellä on runsaasti kokemusta sekä korkeakoulu- että yrityskoulutuksista.

9 789528 026341

BoD™ BOOKS on DEMAND

www.bod.fi