

Webanwendungen

Vorlesung - Hochschule Mannheim

NodeJS + Express

Inhaltsverzeichnis

- ▶ Einführung
- ▶ Architektur/ REST
- ▶ Serving static Files/ Routing
- ▶ Body-Parser
- ▶ Example

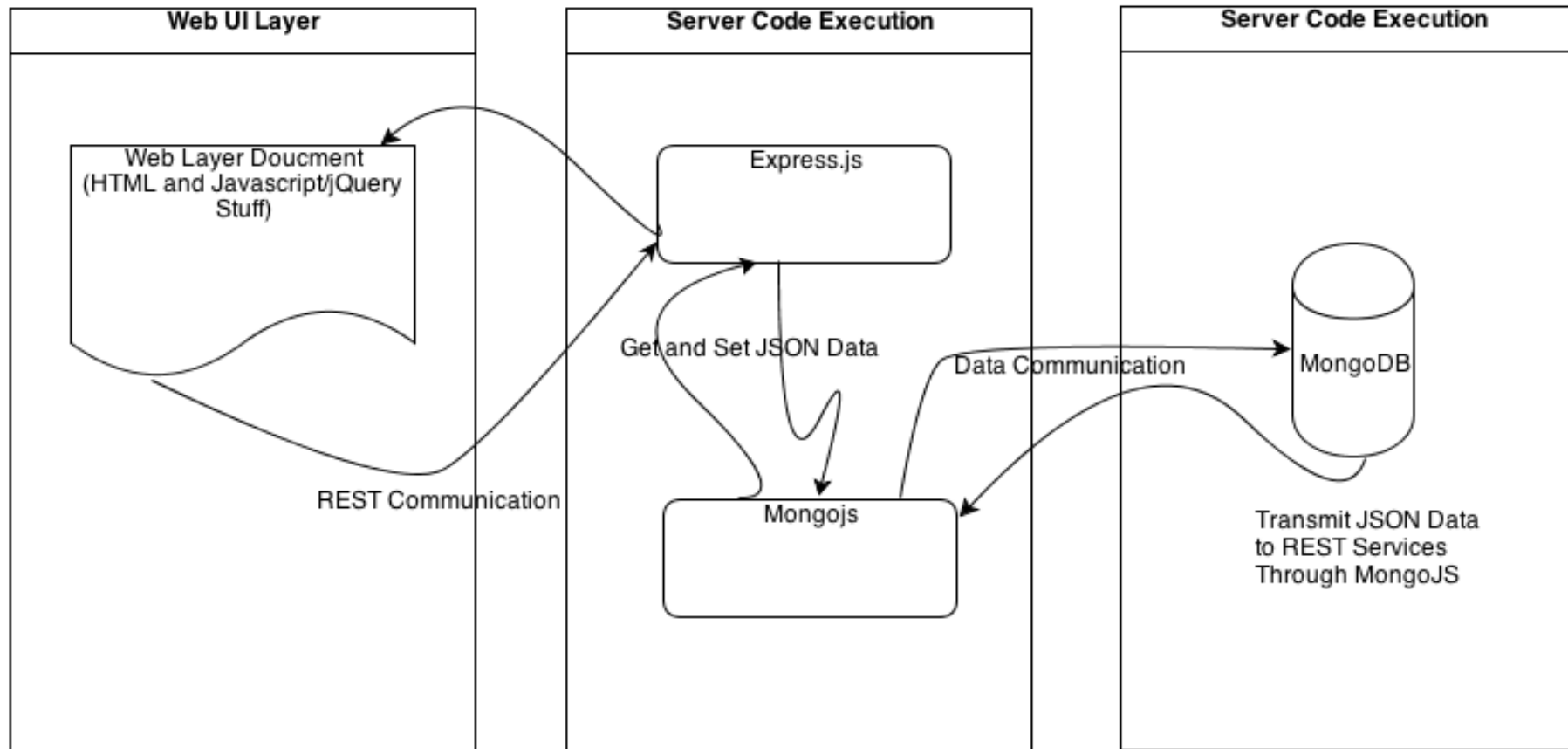
Einführung

ExpressJS



- ▶ Serverseitiges Web Application Framework für Node.js
- ▶ Entwickelt von Douglas Christopher Wilson and community
- ▶ Aktuelle Version: 4.0 (MIT License)
- ▶ Erweitert Node.js an Funktionalität
 - Ausliefern statischer Dateien (CSS/ JS/ HTML)
 - Routing
 - Aufbau des Response

Architektur

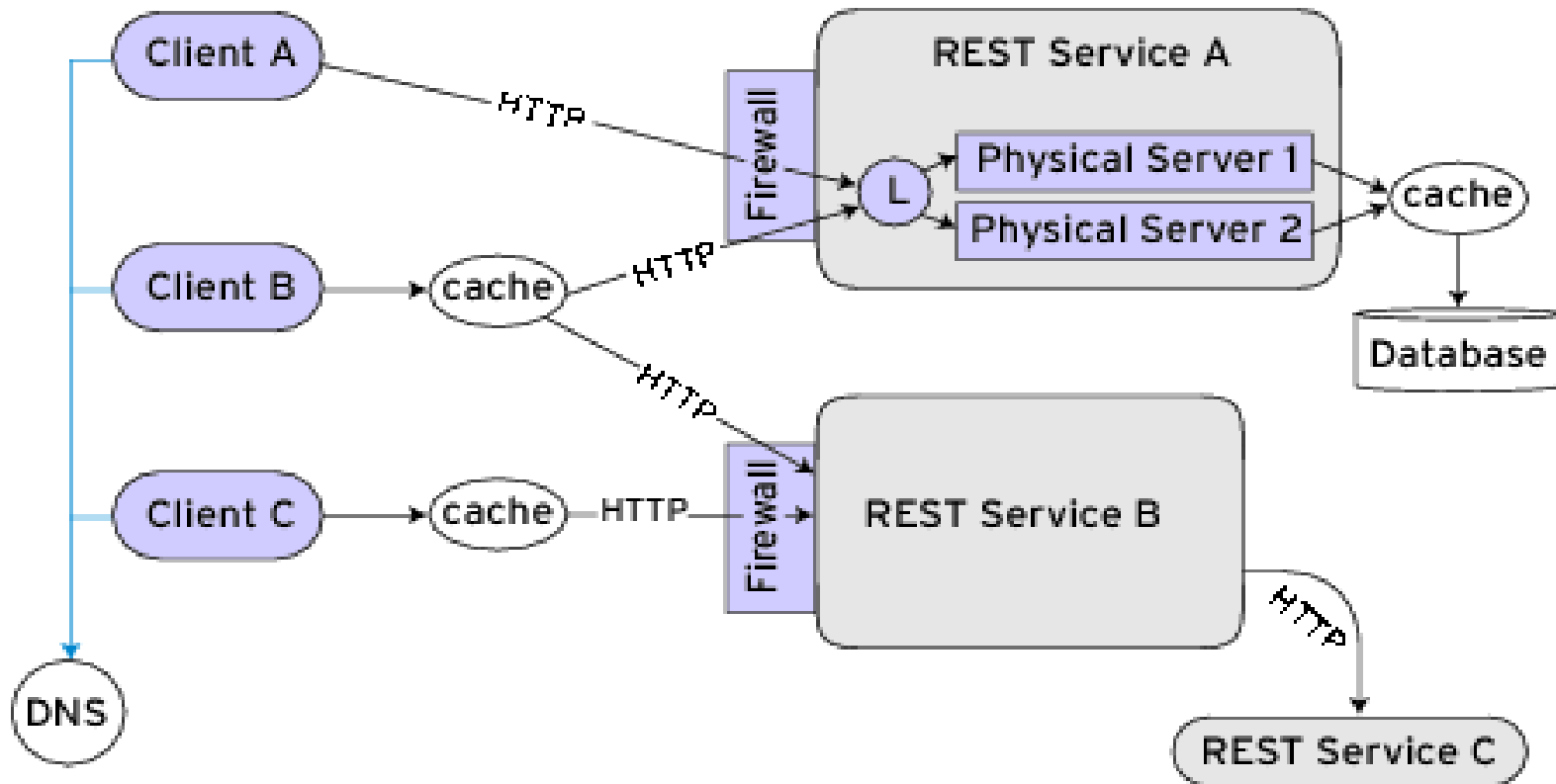


<http://www.phloxblog.in/wp-content/uploads/2013/05/Node-Express-Mongo.png>

REST

- ▶ Representational State Transfer
- ▶ Programmierparadigma für verteilte Systeme
- ▶ Service Orientierte Architektur (SOA)
 - REST ist eine Implementierung von SOA
 - Daten werden als “Services” zur Verfügung gestellt.
 - HTTP dient hierbei als Abfragesprache (bei REST)
 - API ist hierbei URL's die von einem WebServer zur Verfügung gestellt werden

Service Orientierte Architektur



http://twimags.com/ddj/images/article/2007/0706/070601eb01_f1.gif

Service (REST)

- ▶ Dienstleistung eines WebServers
- ▶ Daten werden als Service zur Verfügung gestellt
- ▶ Jeder Dienst (jedes Datum) hat eine eindeutige Adresse
 - Ein Datum kann hierbei eine JSON Datei sein
- ▶ Adressen sind hierbei URIs
 - localhost/blog/42 → Blogartikel mit ID 42
 - localhost/blog → alle Blogartikel
 - localhost/blog/user/11 → Ein User des Blogs mit der ID 11

Service (REST)

- ▶ API Interface sind hierbei HTTP-Methoden
 - GET, PUT, DELETE usw.
 - Alle Operationen lassen sich darauf abbilden.

PUT localhost/blog/42 → legt Blogartikel mit ID 42 an

GET localhost/blog → alle Blogartikel

DELETE localhost/blog/user/11 → löscht User mit der ID 11

Anforderung an WebServer

- ▶ APIs zur Verfügung stellen.
- ▶ Routingsschema der URL != Ordnerstruktur des WebServers
 - Mapping der Routen auf die Ressourcen
 - Client gibt via MIME Type an wie er die Daten haben möchte (JSON, XML)
 - Server muss die Ressourcen dementsprechend umwandeln.
- ▶ Unterstützung aller HTTP Methoden

Bisher mit Node

- ▶ Modul `url` für das Handling und parsen von URLs
- ▶ Oder durch `if(request.url == “/home”)`
- ▶ Das Ausliefern von HTML/ JS und CSS musste eigenständig implementiert werden

Simple WebServer ExpressJS

```
var express = require('express')
var app = express()
app.get('/', function(req, res) {
  res.end('Hello World!')
})
//to set portnumber with terminal
app.listen(process.argv[2])
```

Simple WebServer ExpressJS

- ▶ Laden des Moduls Express in der JavaScript-Server Datei
- ▶ Express nutzt Node.js

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

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

- ▶ Requesthandler wenn ein Request mit der HTTP-Methode GET für die URL localhost/home eingetroffen ist

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

Static Files

- ▶ Style Sheets (CSS), JavaScript Files, Images und HTML-Files
- ▶ Werden nur mit der HTTP-Methode GET aufgerufen
- ▶ Unabhängig der REST-API
 - Da nicht zwangsläufig zur Datenauslieferung benötigt

GET localhost/style.css

GET localhost/index.html

GET localhost/scripts.js

Serving Static Files in Express

- ▶ Konfiguration der Express App

```
app.use(express.static(path.join(__dirname, "public")));
```

- ▶ Mit *app.use* können Express-Parameter zur Konfiguration übergeben werden
- ▶ path: NodeJS Core Modul für Handling von Applikationspfaden.
path = require("path");
path.join([path1][, path2][, ...]) //Konkatenation der strings
- ▶ express.static(<dirname>;
 - Gibt an unter welchem Dateipfad die statischen Dateien zu finden sind. (hier: /applicationRootPath/public/)

Serving Static Files in Express



The screenshot shows a code editor with a file explorer on the left. The file explorer has a tree view with 'express' expanded, showing 'node_modules', 'public', and 'Express.js'. The 'public' folder is expanded, showing 'index.html'. The code editor displays the following JavaScript code:

```
1  var express = require('express');
2  var path = require('path');
3  var app = express();
4
5  app.use(express.static(process.argv[3] || path.join(__dirname, 'public')));
6
7  app.get('/home', function(req, res) {
8    res.end('Hello World!')
9  })
10 //to overgive portnumber
11 app.listen(process.argv[2])
12
```

<http://localhost:3000/index.html>

→ Auslieferung der Datei index.html

<http://localhost:3000/home>

→ “Hello World!”

Basic Routing in Express

`app.METHOD(PATH, HANDLER)`

- ▶ METHOD: eine Http Methode
- ▶ PATH: aufrufender Pfad
- ▶ HANDLER: Requesthandler

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

Basic Routing in Express

POST

- Legt Dateien an
- Daten über Request-Body
- Nicht sicher, nicht idempotent

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

Basic Routing in Express

PUT

- Legt Dateien an
- Daten über Request-Body
- Nicht sicher, aber idempotent

```
app.put('/user', function (req, res) {  
  res.send('Got a PUT request at /user');  
});
```

Basic Routing in Express

DELETE

- Löscht eine Datei
- Nicht sicher, aber idempotent

```
app.delete('/user', function (req, res) {  
  res.send('Got a DELETE request at /user');  
});
```

Basic Routing in Express

- ▶ Query Strings sind nicht Teil des PATH
 - Und sind nicht REST-Konform
- ▶ Regular Expressions sind als Pfad gültig

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

}); → match abcd, abxxd, abRABDOMcd, ab123cd usw.

```
app.get('/ab(cd)?e', function (req, res) {  
  res.send('Hello World!');
```

}); → match /abe und /abcde.

Basic Routing in Express

- ▶ Query Strings sind nicht Teil des PATH
 - Und sind nicht REST-Konform
- ▶ Regular Expressions sind als Pfad gültig

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

```
}); → match abcd, abxxd, abRABDOMcd, ab123cd usw.
```

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

```
}); → match butterfly, dragonfly usw.
```

Basic Routing in Express

Variablen

- ▶ Jede Ressource ist eindeutig über ihre URL identifizierbar

```
app.get('/users/:id?', function(req, res){  
  var id = req.params.id;  
  if (id) {  
    // do something  
  }  
});
```

Response Objekt Funktionen

| Methode | Beschreibung |
|------------------|---|
| res.download() | Veranlasst ein File-Download |
| res.end() | Beendet den Response-Prozess |
| res.json() | Sendet eine JSON-Datei als Response |
| res.redirect() | Redirect einen Request |
| res.render() | Rendert ein Template einer View |
| res.send() | Sendet eine Response (egal welchen Types) |
| res.sendFile | Sendet eine Datei (als Octet Stream) |
| res.sendStatus() | Sendet den ResponseStatus |

Request Body

- ▶ Wichtig bei PUT und POST-Methoden
- ▶ Ist in *req.body* gespeichert
- ▶ Der Request Body beinhaltet nur Key-Value Paare
 - Schlecht auszulesen (kein Objekt)
 - Kein encoding
 - Keine “Konfigurationsmöglichkeiten” des Bodys

Body-Parser

- ▶ Hilfs-Modul für das Parsen des Request-Bodys
 - JSON Body Parser
 - Raw Body Parser
 - Text Body Parser
 - Url-encoded form Parser

Body-Parser

- ▶ Installation

```
$ npm install body-parser
```

- ▶ API

```
var bodyParser = require('body-parser')
```

- ▶ Express Konfiguration zur Nutzung:

```
app.use(bodyParser.json()); // for parsing application/json
```

```
app.post('/profile', function (req, res) {  
  console.log(req.body);  
  res.json(req.body);  
});
```

Body-Parser

bodyParser.json(options)

- ▶ **inflate:** *<boolean:true>*
 - true: komprimierter Body wird dekompromiert
 - false: komprimierter Bodys werden zurückgewiesen
- ▶ **limit:** *<bytes:100kb>* Limitierung der Größe des Bodys
- ▶ **strict** *<boolean: true>*
 - true: akzeptiert nur Arrays und Objekte
 - false: akzeptiert alles

Install Express

- ▶ Mit Hilfe von npm:

```
npm install express
```

- ▶ Zu finden in <projectRoot>/node_modules/express

<http://expressjs.com/en/starter/installing.html>

“Hello World”-Example

```
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);
});
```

\$ node app.js