Die IBS-Backends

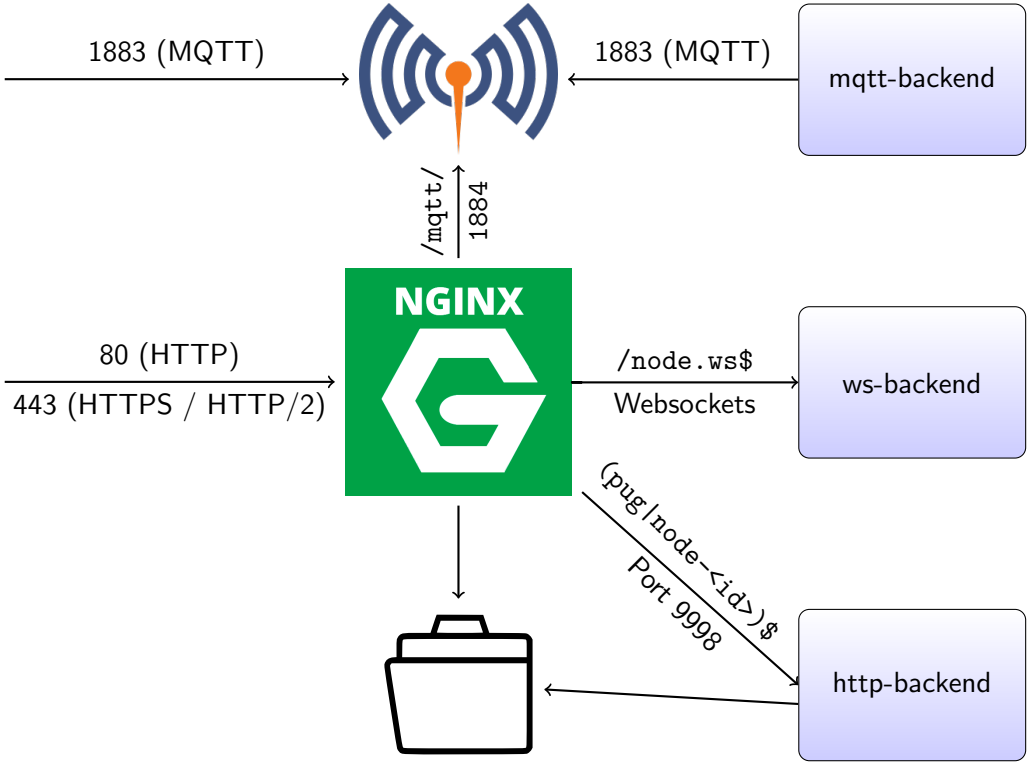
HTTP / FastCGI – Websockets – MQTT

Winfried Bantel

Aalen University

10. Mai 2022

Das Gesamtsystem



HTTP / FastCGI-Backend

Benötigte Pakete

Unbedingt:

express	Webserver-Middleware
pug	Template-Engine
fs	Filesystem (wird mit express mitinstalliert)

FastCGI-Modus:

node-fastcgi	FastCGI-Interface

HTTP-Modus:

serve-index	optional, für Verzeichnis-Listings

Kommandozeilenparameter

1: `mode` `FastCGI` oder anders (typ. `http`)

2: `port` Typ. 80 für http, für FastCGI je nach nginx

3: `html_path` Pfad des HTML-Wurzelverzeichnisses (mit trailing-slash)

Interaktiver Start


```
==> node ibs-backend.js FastCGI 9998 /usr/share/nginx/html/
IBS FastCGI-server listening at port  9998
==>
```

Beenden mit <Strg>-C !

Automatischer Start (1)

Datei /etc/systemd/system/ibs-backend.service:

```
[Unit]
Description=ibs-backend
After=network.target
After=systemd-user-sessions.service
After=network-online.target

[Service]
User=www-data
Group=www-data
Environment=NODE_PATH=/usr/local/lib/node_modules
#WorkingDirectory=/home/wbantel/dummy
ExecStart=/usr/local/bin/node /some/where/http-backend.js FastCGI 9998 "/us
Restart=always

[Install]
WantedBy=multi-user.target
```

Automatischer Start (2)

Ggf. anzupassen:

`Environment` Wenn benötigte node-Module global installiert sind

WorkingDirectory Wenn Module lokal installiert sind

ExecStart Pfad zu node und Pfad zu ibs-backend.js

Start / Stop / restart:

```
==> sudo systemctl start ibs-backend
==> sudo systemctl stop ibs-backend
==> sudo systemctl restart ibs-backend
==> sudo systemctl enable ibs-backend
==> sudo systemctl disable ibs-backend
==>
```

Parameter

- URI-Parameter werden durch express verarbeitet (query-Unterobjekt)
- http-body wird abhängig vom Mime-Type verarbeitet (body-Unterobjekt):

  json falls Mime-Type `application/json`

  http falls Mime-Type `application/x-www-form-urlencoded`

nginx-Konfig


```
location ~ node.js$ {
   return 404;
}

location ~ (\.pug|/node-[A-Za-z_][A-Za-z_0-9]*)$ {
   fastcgi_pass 127.0.0.1:9998;

   fastcgi_param   REQUEST_METHOD          $request_method;
   fastcgi_param   REQUEST_URI             $request_uri;
   fastcgi_param   CONTENT_TYPE            $content_type;
   fastcgi_pass_request_body on;
   proxy_cache off;
   fastcgi_buffering off;
   proxy_buffering off;
   proxy_buffer_size 0;
   gzip off;
}
```

Anpassen: `fastcgi_pass`
Achtung, muss in nginx bei Port 80 und 443 eingetragen sein!

PUG-Schnittstelle

- URI-Pfade die auf pug enden werden per pug gerendert
- Die folgenden Objekte aus http-backend bzw. express werden interpoliert:

req	Von express incl. Body- und Request-Parameter
res	von express
data	für requestübergreifende Daten
plugin	für importierte Libraries (crypto)

JS-Schnittstelle (1)

- URI-Pfade welche auf `node-<id>` enden werden durch das HTTP-Backend verarbeitet
- Dateiname: `node.js`
- Modul muss exportieren: Funktion `<id>`
- `<id>` ist RegExp im Sinne C-Bezeichner:
  `[A-Za-z_][A-Za-z_0-9]*`
- Es erfolgt automatischer Modul-Reload!!!!
  (Kein Server-Neustart notwendig!!!)

Modul-Funktionsaufruf im Backend


```
1       let resp = await modules[mod].mod[req.params.fn](req, res);
2       console.log("resp", typeof resp, resp);
3       if (typeof resp == "object") {
4   //        if (typeof resp.then !== 'function') // Kein promise
5           res.type('json').send(JSON.stringify(resp));
6       }
7       else if (typeof resp == "string")
8         res.send(resp);
9       else if (typeof resp == "number")
10        res.send(String(resp));
11      else if (typeof resp != "undefined")
12        res.end("");
```

JS-Schnittstelle: helloworld

Hochschule Aalen

```
1 exports.helloworld = () => {
2   return "Hello␣world!";
3 }
```

JS-Schnittstelle: helloworld

Hochschule Aalen

```
1 exports.helloworld = () => {
2   return "Hello␣world!";
3 }
```

```
==> curl  -i "localhost/ibs/backend/node-helloworld"
HTTP/1.1 200 OK
Server: nginx/1.14.0
Content-Type: text/html; charset=utf-8
Content-Length: 12
Connection: keep-alive
X-Powered-By: Express
ETag: W/"c-00hq6RNueFa8QiEjhep5cJRHWAI"
Set-Cookie: connect.sid=s%3Almhvt5AeH6U6qKcv6MsynM2...; Path=/; HttpOnly
Date: Mon, 25 Apr 2022 07:25:15 GMT

Hello world!%
==>
```

JS-Schnittstelle: Objektrückgabe


```
1 exports.time = () => {
2   return {time: new Date()};
3 };
```

JS-Schnittstelle: Objektrückgabe


```
1 exports.time = () => {
2   return {time: new Date()};
3 };
```

```
==> curl  -i "localhost/ibs/backend/node-time"
HTTP/1.1 200 OK
Server: nginx/1.14.0
Content-Type: application/json; charset=utf-8
Content-Length: 35
Connection: keep-alive
X-Powered-By: Express
ETag: W/"23-yNJXdCKOCJkc/UPmqFLVIM5mwK0"
Set-Cookie: connect.sid=s%3ADdRSczh11eapgwpjYU0F...; Path=/; HttpOnly
Date: Mon, 25 Apr 2022 07:30:25 GMT

{"time":"2022-04-25T07:30:25.041Z"}%
==>
```

JS-Schnittstelle: Zahlenrückgabe


```
1 exports.unix_time = () => {
2   return (Date.now() / 1000) >> 0;
3 }
```

JS-Schnittstelle: Zahlenrückgabe


```
1 exports.unix_time = () => {
2   return (Date.now() / 1000) >> 0;
3 }
```

```
==> curl  -i "localhost/ibs/backend/node-unix_time"
HTTP/1.1 200 OK
Server: nginx/1.14.0
Content-Type: text/html; charset=utf-8
Content-Length: 10
Connection: keep-alive
X-Powered-By: Express
ETag: W/"a-5cWEBRx2h+LLTlI5RS2pezY7Els"
Set-Cookie: connect.sid=s%3A208AOX2ALNZIWV5vGlFHt...; Path=/; HttpOnly
Date: Mon, 25 Apr 2022 07:32:07 GMT

1650871927%
==>
```

JS-Schnittstelle: request-Objekt


```
1 exports.req_object = (req) => {
2   return JSON.stringify({
3     method: req.method,
4     "content-type": req.header("content-Type"),
5     "req.query": req.query,
6     "req.body": req.body,
7   } , null, 4) + "\n";
8 }
```

JS-Schnittstelle: result-Objekt

Hochschule Aalen

```
1  exports.res_object_1 = (req, res) => {
2    res.type("text/plain");
3    res.send("Hallo␣Welt!\n")
4  }
```

```
1  exports.res_object_2 = (req, res) => {
2    res.type("text/plain");
3    for (let i = 1; i <= 10; i++)
4      res.write(`${i} * ${i} = ${i * i}\n`);
5    res.end();
6  }
```

JS-Schnittstelle: Globale / dauerhafte Variablen


```
1 var counter_nr = 0;
2 exports.counter = () => {
3   return `Hallo ${++counter_nr}`;
4 };
```

JS-Schnittstelle: Globale / dauerhafte Variablen


```
1 var counter_nr = 0;
2 exports.counter = () => {
3   return `Hallo ${++counter_nr}`;
4 };
```

```
==> for i in {1..10}
do
curl "localhost/ibs/backend/node-counter"
echo ""
done
Hallo 24
Hallo 25
Hallo 26
Hallo 27
Hallo 28
Hallo 29
Hallo 30
Hallo 31
Hallo 32
Hallo 33
==>
```

JS-Schnittstelle: Asynchrone Antwort


```
1 exports.slow_1 = (req, res) => {
2   setTimeout(() => {
3     res.send(String(Math.random()));
4   }, 1000);
5 }
```

JS-Schnittstelle: Asynchrone Antwort


```
exports.slow_1 = (req, res) => {
  setTimeout(() => {
    res.send(String(Math.random()));
  }, 1000);
}
```

```
exports.slow_2 = () => {
  return new Promise(r =>
    setTimeout(() => r(Math.random()), 1000)
  );
}
```

JS-Schnittstelle: Asynchrone Antwort


```
exports.slow_1 = (req, res) => {
  setTimeout(() => {
    res.send(String(Math.random()));
  }, 1000);
}
```

```
exports.slow_2 = () => {
  return new Promise(r =>
    setTimeout(() => r(Math.random()), 1000)
  );
}
```

```
exports.slow_3 = async () => {
  await new Promise(r => setTimeout(() => r(), 1000));
  return Math.random();
}
```

Websocket-Backend

Kommandozeilenparameter

1: `port` Typ. 80 für http, für FastCGI je nach nginx

2: `html_path` Pfad des HTML-Wurzelverzeichnisses (mit trailing-slash)

nginx-Konfig


```
location ~ /node.ws$ {
            proxy_pass http://127.0.0.1:9996;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_read_timeout  36000s;
}
```

Anpassen: `proxy_pass`
Achtung, muss in nginx bei Port 80 und 443 eingetragen sein!

Interaktiver Start


```
==> node ws-backend.js  9996 "/path/to/html"
==>
```

Beenden mit <Strg>-C !

Automatischer Start (1)

Datei /etc/systemd/system/ibs-backend.service:

```
[Unit]
Description=ibs-backend
After=network.target
After=systemd-user-sessions.service
After=network-online.target
[Service]
User=www-data
Group=www-data
#Environment=NODE_PATH=/usr/local/lib/node_modules
WorkingDirectory=/media/sf_websocketfreigabe/ibs-backend/
ExecStart=/path/to/node /path/to/ws-backend.js  9996 "/path/to/html"
Restart=always
StandardOutput=file:/tmp/httpbackend-stdout.log
StandardError=file:/tmp/httpbackend-stderr.log
[Install]
WantedBy=multi-user.target
```

Pfade

- URI-Pfad des ws-Verbindungsaufbaus wird als Directory erwartet
  (Unter Server-HTML-Directory)
- Im Pfad muss Datei `node-ws.js` extistieren
- `node-ws.js` muss als Modul drei Funktionen exportieren:

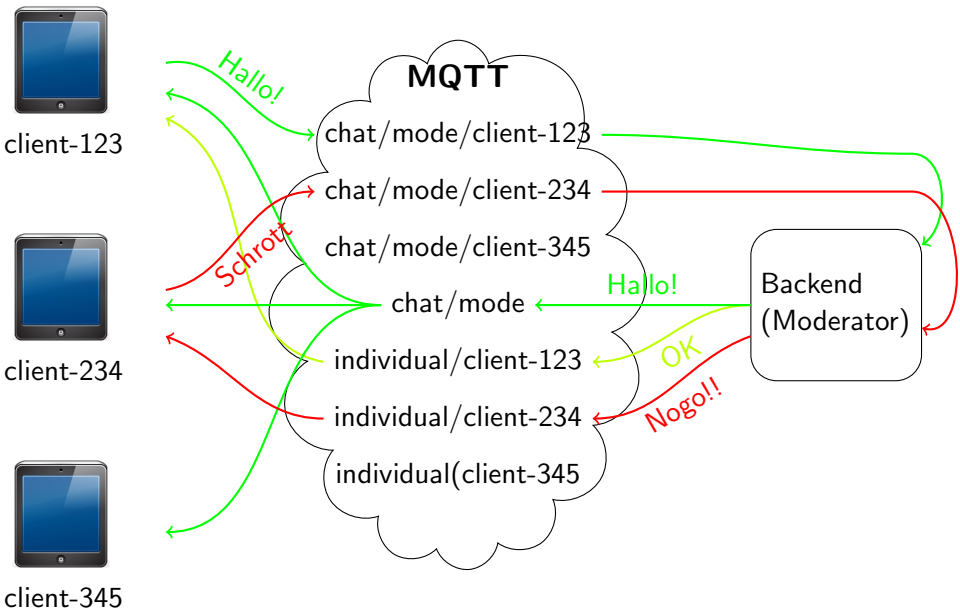
  `connection` wird aufgerufen wenn eine ws-Verbindung beginnt

  `message` wird aufgerufen wenn für eine ws-Verbindung eine NAchricht eingeht

  `close` wird aufgerufen wenn eine ws-Verbindung endet

MQTT-Backend

Fallbeispiel: Moderierter Chat



Kommandozeilenparameter

`prm` Pfad zur Parameterdatei

Parameterdatei

(Exemplarisch...)

```
{
   "host": "localhost",
   "port": 1883,
   "conn_prm": {},
   "topics": [
       {"topic": "mqtt-backend/+/2", "mod": "abc", "f": "f"},
       {"topic": "dummy/#", "mod": "./abc", "f": "g"}
   ]
}
```

Topic-Eintrag:

- topic: Zu subscribierendes Topic (MQTT-Wildcards erlaubt)
- mod: JS-Moduldatei (ohne Namenserweiterung). Absoluter Pfad oder relativ ab working-directory
- f: Funktion des Moduls

Schnittstelle aufgerufene Funktion

Parameter:

1. Topic
2. Payload
3. Packet (Zusatzinfo)

Rückgabe: Einzelnes Objekt oder Array aus Objekten der Struktur

```
{
    topic: "hallo",
    payload: "welt"
}
```

nginx config

(ggf anpassen)

```
1
2           location =  /mqtt/ {
3               proxy_pass http://127.0.0.1:1884;
4               proxy_http_version 1.1;
5               proxy_set_header Upgrade $http_upgrade;
6               proxy_set_header Connection "upgrade";
7               proxy_read_timeout  36000s;
8           }
```