FastCGI for YottaDB - Installation and Quick-Start

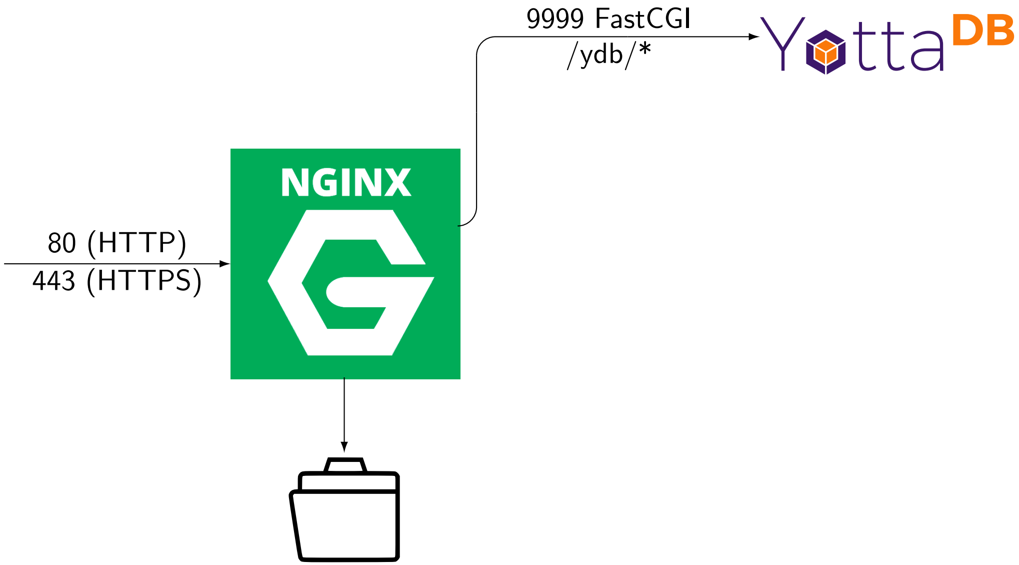
Winfried Bantel

Aalen University

6. April 2019

Aalen University

The nginx-system



Advantages NOT to write a HTTP-Server

- Very very fast FastCGI-backend written in native M-language
- Runs on YottaDB and GT.M
- nginx is able to cache - less work for M-Backend
- HTTPS supported by nginx
- HTTP/2 supported by nginx
- HTTP/2 with dynamic server push for even faster applications
- Filebased Webserver is done by nginx
- With JSON-Parser ideal backend for Single-Page-Applications (i.e. with AngularJS)
- Supports massive parallel HTTP-requests
- Sensible data can be stored physically on another machine
- Other backends like php, couchdb on the same webserver

Why is FastCGI so fast?

- The connection between the WWW-server and the FastCGI-backend remains open
- So no time needed to start jobs and so on (from the second request on)
- The FastCGI-backend rests listening on an IP-Port
- The WWW-server acts nearly like a TCP/IP-router

Installation-Steps

You should be firm in YottaDB!

1. Install nginx
2. Edit nginx-Config
3. Install YottaDB
4. Install xinetd
5. Edit xinetd-Config-Script
6. Copy FCGI.m
7. Set a global
8. Be happy

User and so on

- In these slides the user is wbantel.
- His home-directory is /home/wbantel/
- If You want another user: adapt!
- YottaDB-distribution is here `/usr/local/lib/yottadb/r124/`
- If You want another distribution: adapt!
- YottaDB uses IP-Port 9999
- If You want another port: adapt nxginx-config and xinetd-config

Step 1: Install nginx

Hochschule Aalen

```
>>> sudo apt install nginx
>>> curl localhost
```

Or test from any Computer in WWW / LAN with IP-Address oder DNS

Step 2: Edit nginx-Config

Hochschule Aalen

```
# minimal nxinx-config for YottaDB-FastCGI
upstream ydb_fcgi_backend {
        server 127.0.0.1:9999;
        keepalive 32;
}
server {
        listen              80;
        listen      [::]:80;
        server_name     localhost;
        root /usr/share/nginx/html/ ;
        index index.html index.htm index.xhtml ;
        location /ydb/ {
                fastcgi_pass ydb_fcgi_backend;
                fastcgi_keep_conn on ;
                fastcgi_param   QUERY_STRING            $query_string;
                fastcgi_param   SID                     $cookie_sid;
                fastcgi_param   DOCUMENT_URI            $document_uri;
                fastcgi_param   REQUEST_METHOD          $request_method;
                fastcgi_param   REMOTE_ADDR             $remote_addr;
        }
}
```

Step 2: Edit nginx-Config

- This is a minimal nginx-config
- Feel free to config HTTPS, HTTP/2, other backends like php and so on

Step 3: Install YottaDB

Have a look at the YottaDB-Website

Step 4: Install xinetd

```
>>> sudo apt install xinetd
>>>
```

Step 5: Edit xinetd-Config-Script

```
 1 service ydb-fastcgi
 2 {
 3         protocol    = tcp
 4         port        = 9999
 5         type        = UNLISTED
 6         socket_type = stream
 7         wait        = no
 8         user        = wbantel
 9         grou        = wbantel
10         server      = /usr/local/lib/yottadb/r124/mumps
11         server_args = -run FCGI
12         env         = ydb_dir=/home/wbantel/.yottadb ydb_gbldir=/home
13         disable     = no
14 }
```

- Adapt lines 8 to 12 to your system!
- To do a little setup-test:
  - change server_args to `-run DOLLARH^FCGI`
  - Don't forget to restart xinetd
  - Test with telnet localhost 9999
  - Redo the changes and restart xinetd

Step 6: Copy FCGI.m

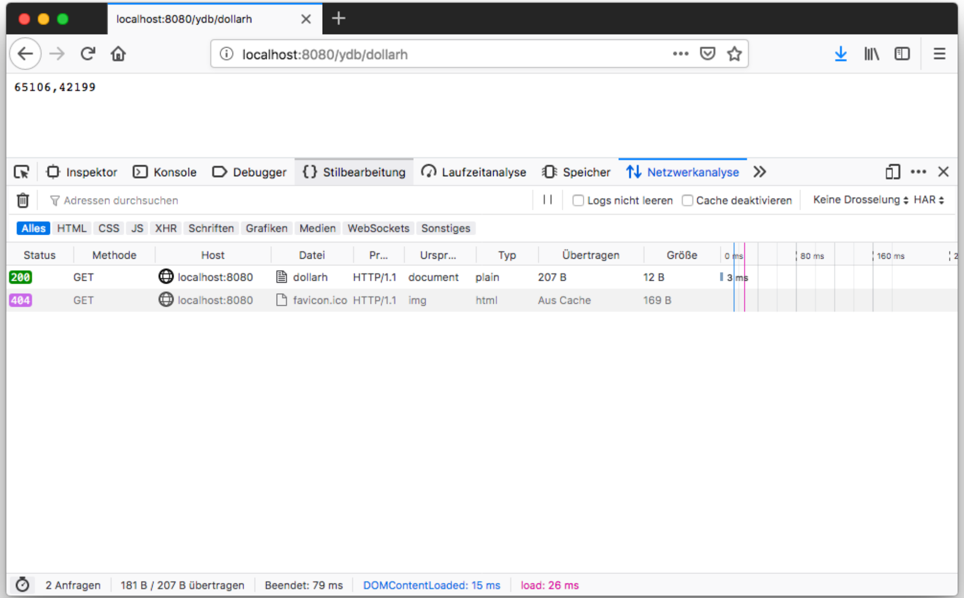
Hochschule Aalen

```
>>> cp /from/some/where/FCGI.m /home/wbantel/.yottadb/r.../r/
>>>
```

Step 7: Set a global

```
>>>  ydb
YDB> SET ^FCGI("DOCUMENT_URI","/ydb/dollarh")="DOLLARH"
YDB>
>>>
```

Step 8: Be happy



localhost:8080/ydb/dollarh

```
65106,42199
```

Status	Methode	Host	Datei	Pr...	Urspr...	Typ	Übertragen	Größe	0 ms 80 ms 160 ms
200	GET	localhost:8080	dollarh	HTTP/1.1	document	plain	207 B	12 B	3 ms
404	GET	localhost:8080	favicon.ico	HTTP/1.1	img	html	Aus Cache	169 B	

2 Anfragen | 181 B / 207 B übertragen | Beendet: 79 ms | DOMContentLoaded: 15 ms | load: 26 ms

Some configuration-parameters

```
^FCGI("PRM","ZLINK")
^FCGI("PRM","LOG")
^FCGI("PRM","GZ")
^FCGI("PRM","TO")
```

ZLINK Use this parameter for developing (set to 1) so when you edit a routine and save it the changes will have an effect (suitable for developing). Otherwise kill the global and it will run a little bit faster (suitable for production).

- 0 (or killed): The called routine will be called without ZLINK
- 1: The called routine will be ZLINKed before called

LOG Some logging in /tmp/fastcgi.log

- 0 (or killed): Logging off
- 1: Logging on

GZ Output written to %fcgi will be compressed before sent. Needs some time, but transmission will be faster. (Not needed for HTTP/2!)

- 0 (or killed): ZIPping off
- 1: ZIPping on

TO Timeout a job will wait for a second request. Default is 60 seconds.

How to call an M-routine

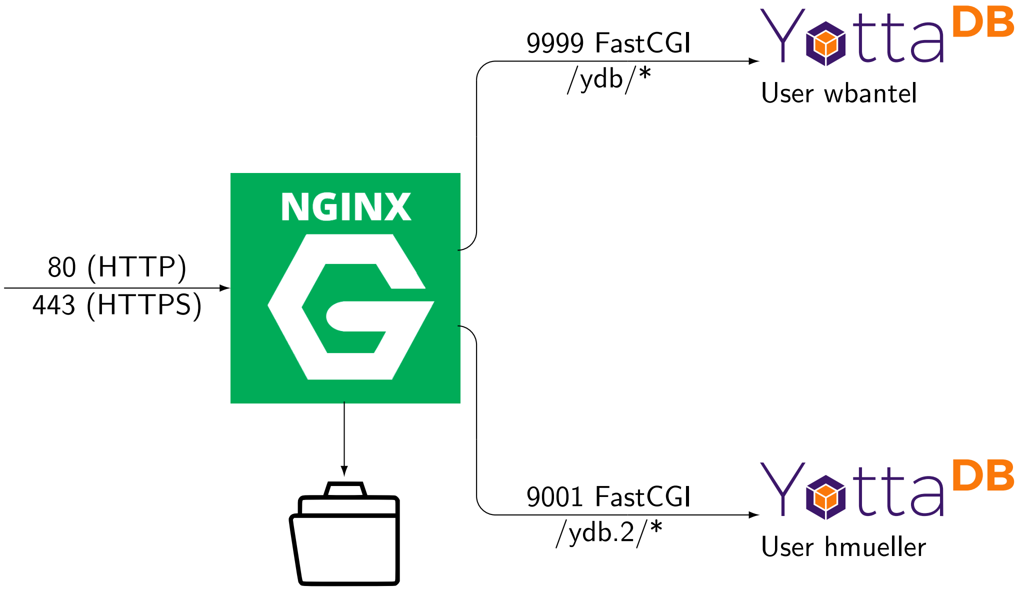
- Indirection-Global is `^FCGI("DOCUMENT_URI",<uri>)`
- FastCGI examines `$PIECE(uri,"/",1,3)`
  Attention, first piece is alway empty! I.e.
  `/ydb/dollarh`
  third piece is dollarh
- Second piece has to be the location from nginx-config-file (usually „ydb“)
- Third piece is variable and used for distribute to application-routine
- Set an Indirection-Global for Your app (see step 7) which points
  `$PIECE(uri,"/",1,3)` → M-code-entryreference
  Example: `^FCGI("DOCUMENT_URI","/ydb/test")="^TEST"`
- Forth / fifth / ... piece can be used in application, i.e. a REST-Interface:
  `/ydb/rest/customer/1` points to rest-interface for file (global) „customer“ and database-index 1
  Therefore examine `%fcgi("i","header","DOCUMENT_URI")`

More than one System (1)

You need / have another YottaDB-System, perhaps for development and production, totally different?

- Create another user
- Create another xinetd-Config with another TCP/IP-Port and another name
- Create another Upstream-Part with the correct Port in nginx-Config
- Create another Location-Part with another URI an the correct Upstream in nginx-Config

More than one System (2)



How to generate Output

Several ways for backend-routine to generate output

1. Write to device %fcgi
2. Set a global-name
3. Set a filename
4. Set a single variable
5. Set an array variable
6. Callback-Functions (direct output)

Don't mix it up, use only exactly one way!

Generate output - device %fcgi

- Easiest way to generate Output

```
EXOUTPUT1	; Generates output using %fcgi
	; On start %fcgi is open and used!!!
	w "<html><head></head><body>",$H,"</body></html>"
```

Generate output - globalbased

- Ideal in case of the global already exists

```
EXOUTPUT2    ; Generates output using global
        s ^dummy="<html><head></head><body>"_$H_"</body></html>"
        s %fcgi("o","glo")="^dummy"
```

Generate output - filebased

- Ideal in case of the file already exists

```
EXOUTPUT3    ; Generates output using file / filename
        s f="/tmp/"_$j_".html" o f:newversion
        u f w "<html><head></head><body>"_$H_"</body></html>" c f
        s %fcgi("o","file")=f
```

Generate output using local variable

```
EXOUTPUT4    ; using local variable
        s %fcgi("o","stdout")="<html><head></head><body>"_$H_"</body>
```

Generate output using local array

```
EXOUTPUT5    ; Generate output using array
        s %fcgi("o","stdout",1)="<html>"
        s %fcgi("o","stdout",2)="<head></head>"
        s %fcgi("o","stdout",3)="<body>"_$H_"</body>"
        s %fcgi("o","stdout",4)="</html>"
```

Generate output using callback-functions

- Fastest of all
- No buffer!

How-to:

1. Set Header (optional)
2. Call `HEADEROUT^FCGI`
3. Call (repeatedly) `DATAOUT^FCGI(...)` (optional, output can be empty)
4. `SET %fcgi("o","noout")=1`

(Have also a look at the html5-sse-example)

Generate output using callback-functions

```
EXOUTPUT6 ; Direct Output

        ; Step 1: optional
        s %fcgi("o","header","Content-Type")="text/plain"

        ; Step 2: mandatory
        d HEADEROUT^FCGI

        ; Step 3: optional
        f i=1:1:5 d DATAOUT^FCGI("Line "_i_$C(10,13)) h 1

        ; Step 4: mandatory
        s %fcgi("o","noout")=1

        q
```

Output-Header

- For Content-Type, Redirect, cookies and so on

```
EXSETHEADER      ; Generates output using %fcgi
        s %fcgi("o","header","Content-Type")="application/json"
        s %fcgi("o","header","X-greeting")="Hello from YottaDB!"
        s %fcgi("o","header","X-HOROLOG")=$H
        w "{""$H"":"""_$H_""",""$J"":"""_$J_"""}"
```

```
>>> curl "localhost/ydb/exsetheader" -i
HTTP/1.1 200 OK
Server: nginx/1.14.0
Date: Fri, 05 Apr 2019 06:49:34 GMT
Content-Type: application/json
Content-Length: 34
Connection: keep-alive
X-HOROLOG: 65108,31774
X-YDB-nr: 1
X-greeting: Hello from YottaDB!
X-job: 22629
X-version: 20190321

{"$H":"65108,31774","$J":"22629"}
```

Session-ID is the new $JOB

Hochschule Aalen

- Session-tracking is forced calling `SID^FCGI`
- Stored in `%fcgi("i","header","SID")`
- Two Comma-separated integers:
  1. 64-bit random-int which is constant for your session
  2. Counter auto-incrementing with each HTTP-request
- Is done by a temporary (non-persistant) cookie
- Ideal for storing session-specific data

```
EXSID    ; Generates output using %fcgi
         q:'$$SID^FCGI()   s sid=%fcgi("i","header","SID")
         w "<html><head></head><body>"
         w "Your Session-ID is ",+sid,"<br>",!
         w "Your Session-count is ",$P(sid,",",2),"<br>",!
         w "Your last visit ($H) was: ",$G(^dummy(+sid)),"<br>",!
         s h=$H w "Now $H is: ",h,"<br>",!
         s ^dummy(+sid)=h
         w "<br>Feel free to reload!"
         w "<br><a href=""javascript:location.reload()"">Reload</a>"
         w "</body></html>"
```

HTTP-GET-vars

- Easiest way to get data from Webclient
- Data is part of the uri:
  `/ydb/something?firstname=Winfried&lastname=bantel`

```
EXGETVAR     ;
        w "<html><head></head><body>"
        i $G(%fcgi("i","_GET","name"))="" d
        . w "You did't enter a name"
        e  w "Hello ",%fcgi("i","_GET","name"),"!"
        w "<form method=""GET"">",!
        w "<input type=""text"" name=""name"">",!
        w "<input type=""submit"" value=""Submit"">",!
        w "</form></body></html>"
```

HTTP-POST-vars

- Better way to get form-data from Webclient
- Data is sent in the HTTP-body

```
EXPOSTVAR	;
	w "<html><head></head><body>"
	i $G(%fcgi("i","_POST","name"))="" d
	. w "You did't enter a name"
	e  w "Hello ",%fcgi("i","_POST","name"),"!"
	w "<form method=""POST"">",!
	w "<input type=""text"" name=""name"">",!
	w "<input type=""submit"" value=""Submit"">",!
	w "</form></body></html>"
```

HTTP-BODY

- Suitable for JSON-data, File-Uploads and so on

```
EXSTDIN     ;
        ; > curl ip-address:port/ydb/EXPOSTVAR -d "Hallo Welt!"
        ; > curl ip-address:port/ydb/EXPOSTVAR -d @file.txt
        ; Or a Browser-form with method post:
        ; <form action="/ydb/EXPOSTVAR" method="POST">...</form>
        w "<html><head></head><body>Your Post-Data is<pre>"
        w $G(%fcgi("i","stdin"))
        w "</pre></body></html>",!
```

```
>>> curl -i "localhost:8080/ydb/EXSTDIN" -d '{"NN":"Bantel"}'
HTTP/1.1 200 OK
Server: nginx/1.14.0
Date: Wed, 09 Jan 2019 14:13:28 GMT
Content-Length: 83
Connection: keep-alive
X-job: 2699
X-nr: 2

<html><head></head><body>Your Post-Data is<pre>{"NN":"Bantel"}</pre></body
```

Complete HTTP-Info

- The complete info is stored in %fcgi

```
EXHTTPINFO    ;;
        s %fcgi("o","header","Content-Type")="text/plain"
        zwr %fcgi
```

```
>>>  curl "localhost:8080/ydb/EXHTTPINFO?test=1"  -d '{"NN":"Bantel"}'
%fcgi="/tmp/fcgi-fifo-4011" ;*
%fcgi("i","FCGI_KEEP_CONN")=1
%fcgi("i","_GET","test")=1
%fcgi("i","_POST","{""NN"":""Bantel""}")=""
%fcgi("i","header","DOCUMENT_URI")="/ydb/EXHTTPINFO"
%fcgi("i","header","HTTP_ACCEPT")="*/*"
%fcgi("i","header","HTTP_CONTENT_LENGTH")=15
%fcgi("i","header","HTTP_CONTENT_TYPE")="application/x-www-form-urlencoded
%fcgi("i","header","HTTP_HOST")="localhost:8080"
%fcgi("i","header","HTTP_USER_AGENT")="curl/7.51.0"
%fcgi("i","header","QUERY_STRING")="test=1"
%fcgi("i","header","REMOTE_ADDR")="10.0.2.2"
%fcgi("i","header","REQUEST_METHOD")="POST"
%fcgi("i","header","SID")=""
%fcgi("i","stdin")="{""NN"":""Bantel""}"
%fcgi("internal","entryRef")="^EXHTTPINFO"
```

Embedding YottaDB-output in static HTML

- Use Server-Side includes
- in nginx-config

```
location /some/where {
        ssi on;
}
```

- in HTML

```
<html>
<head></head>
<body>
<h1>YottaDB</h1>
<pre><!--# include virtual="/ydb/EXHTTPINFO?$args" --></pre>
</body>
</html>
```

- Works only for HTTP-GET

YottaDB on another server

A WWW-Server is always vulnerable!
Secure Your important data, don't store them in the WWW-Server!

- Change the address of the FastCGI-backend in nginx-config (see Step 2 above)!
  Example:

  ```
  upstream ydb_fcgi_backend {
          server 192.168.10.12:9999;
          keepalive 32;
  }
  ```

- Run YottaDB and xinetd at 192.168.10.12

Load balancing: YottaDB on multiple servers

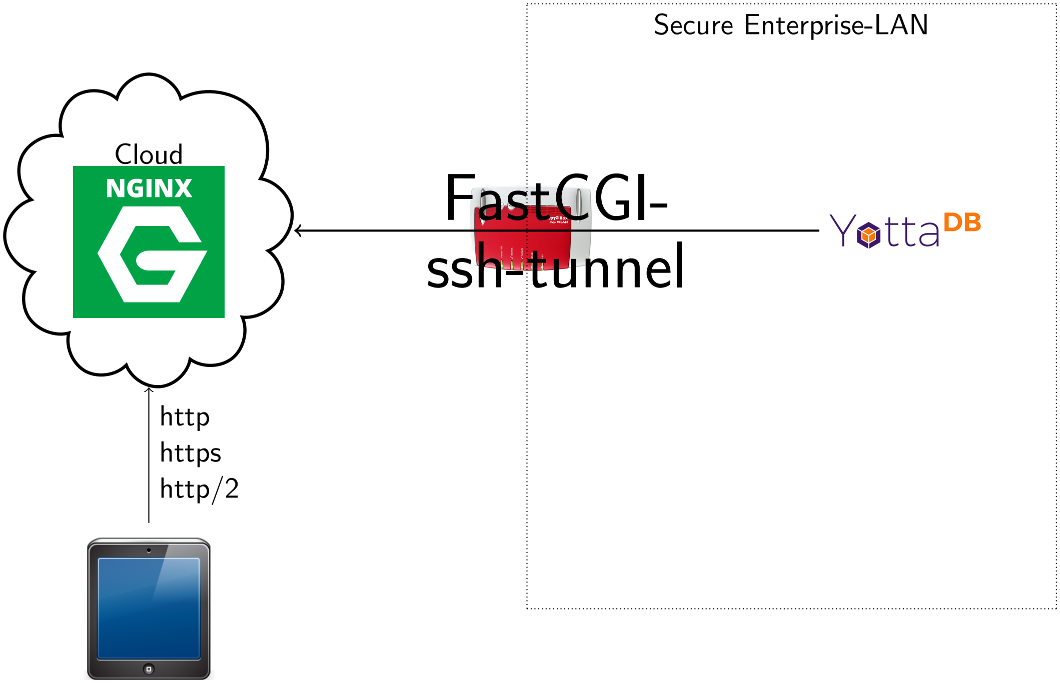
Hochschule Aalen

- More than one FastCGI-backend in nginx-config (see Step 2 above)!
  Example:

```
upstream ydb_fcgi_backend {
        server 192.168.10.12:9999;
        server 192.168.10.13:9999;
        server 192.168.10.14:9999;
        server 192.168.10.15:9999;
        keepalive 32;
}
```

- Run YottaDB and xinetd at 192.168.10.12 ... 192.168.10.15
- The servers should exactly be mirrored!

YottaDB behind a firewall (1)



YottaDB behind a firewall (2)

- M is in an enterprise-LAN
- nginx is somewhere in the WWW
- A Firewall whithout Port-Forwarding
- With ssh-tunnel M-Backend becomes a TCP/IP-client in the LAN

To enable start in the M-Server

```
ssh -Nf -R 9999:localhost:9999 www.my-web-server.de
```

(Can be done better with autossh)

YottaDB as an AJAX-backend

- HTML is a static file
- Browser polls event-triggered information
- Attention, for this example the YottaDB-redirect-uri `/ydb/dollarh` must be set the same as in example-line-nr 6 (see installation-step 7)
- In this example the event is a timer (polling):

  ```
  window.setInterval(get_data, 1000);
  ```

YottaDB as an AJAX-backend

```
<html>
<head><title>YottaDB via AJAX</title>
<script language="JavaScript">
        function get_data(){
                var _http = new XMLHttpRequest;
                _http.open("GET","/ydb/dollarh", true);
                _http.onreadystatechange = function() {
                        if (_http.readyState === 4)
                                document.getElementById("ausgabe").firstChild.
                                        nodeValue = _http.responseText;
                };
                _http.send();
        }
</script>
</head>
<body onload="window.setInterval(get_data,1000)">
<h1>$HOROLOG by AJAX</h1>
$H: <output id="ausgabe"> </output>
</body>
</html>
```

Using HTTP/2

- HTTP/2 is the future
- Uses
  - Compression (even for the HTTP-head)
  - Encryption
  - Persistant connections
  - Parallel connections
- nginx supports http/2
- nginx supports dynamic server push for FastCGI-backends
- How to use HTTP/2?
  Configure nginx - that's all! There is nothing to change for YottaDB-FastCGI!

```
>>> curl --http2 -ik "https://localhost/ydb/dollarh"
HTTP/2 200
server: nginx/1.14.0
date: Fri, 05 Apr 2019 06:53:31 GMT
content-type: text/plain
content-length: 12
x-ydb-nr: 1
x-job: 22655
x-version: 20190321

65108,32011
```

Using HTTP/2 server-push

- With HTTP/2 there can be sent more than one document for one request
- In example:
  - A HTML-page with an img-tag
  - The static image for the image-tag
- It is much faster than loading html, parsing, loading image
- How to do? Jus set a HTTP-header, nginx will do the rest!

1. In the nginx-config (minimum version 1.13.9)

```
     location /ydb/ {
         http2_push_preload on;
        ...
    }
```

2. In the M-backend-program: Set HTTP-Header „Link“:

```
s %fcgi("o","header","Link")="</ibs/http-2/server-push.css>; rel=preloa
w $J_" "_$H_" "_$IO
```

For details visit
`https://www.nginx.com/blog/nginx-1-13-9-http2-server-push/`

Single-Page-Apps with AngularJS

- Modern Web-2.0-application
- Static HTML – delivered by nginx
- Backend (YottaDB) serves JSON-data
- Attention, the YottaDB-redirect-uri has to be the same as variable uri in exangularjs.js line-nr 4!

Single-Page-Apps with AngularJS – HTML

Hochschule Aalen

```
<!doctype html>
<html ng-app="ajaxApp">
 <head><title>AngularJS with YottaDB</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.
  <!--<script src="/lib/angular-1-7-8.min.js"></script>-->
  <script src="exangularjs.js"></script>
 </head>
 <body ng-controller="Controller as q">
  <table>
   <tr><th>ID:</th><td>
    <input size="3" ng-model="q.id"/>
    <input type="button" value="Load" ng-click="q.load()"/>
   </td></tr>
   <tr><th>Vorname:</th><td>
    <input type="text" ng-model="q.address.VN">
    <tr><th>Nachname:</th><td>
    <input type="text" ng-model="q.address.NN"></th></tr>
   <tr><th></th><td>
    <input type="button" value="Save" ng-click="q.send()"/>
   {{q.savetext}}</td></tr>
  </table>
 </body>
```

Single-Page-Apps with AngularJS – JS

Hochschule Aalen

```
var app = angular.module('ajaxApp', []);
app.controller('Controller', function($scope, $http) {
    var c = this;
    var uri = "/ydb/EXANGULARJS/";

    c.send = function() {
        $http.put(uri+c.id,c.address).then(function (response) {
            c.savetext = JSON.stringify(response.data);
            setTimeout(function(){
                c.savetext = ""; $scope.$apply();
            }, 2500);
        });
    };

    c.load = function() {
        $http.get(uri+c.id).then(function (response) {
            c.address =(response.data);
        });
    };
});
```

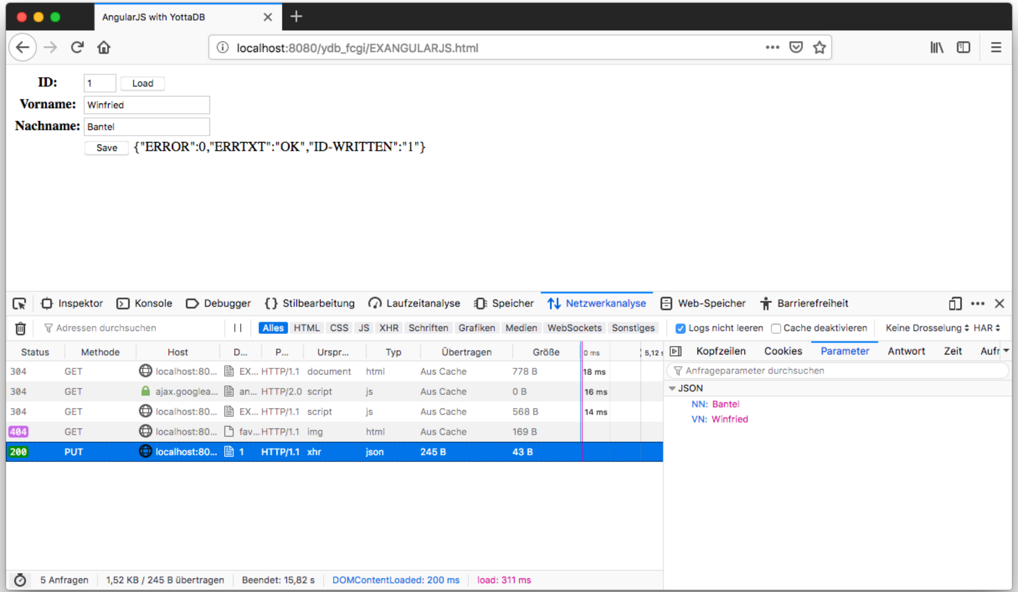
Single-Page-Apps with AngularJS – M

```
EXANGULARJS   ; A very very simple REST-Interface
        s %fcgi("o","header","Content-Type")="application/json"
        s id=+$P(%fcgi("i","header","DOCUMENT_URI"),"/",4)
        i id<=0 w "{""ERROR"":1}" q

        i %fcgi("i","header","REQUEST_METHOD")="PUT" d
        . s ^EXANGULARJS(id)=%fcgi("i","stdin")
        . w "{""ERROR"":0,""ERRTXT"":""OK"",""ID-WRITTEN"":""" _id_"""
        e  d
        . w $S($D(^EXANGULARJS(id)):^(id),1:"{}")
```

```
>>> curl "localhost/ydb/EXANGULARJS/123"
{}
>>> curl -X PUT "localhost/ydb/EXANGULARJS/123" -d '{"NN":"Bantel"}'
{"ERROR":0,"ERRTXT":"OK","ID-WRITTEN":"123"}
>>> curl "localhost/ydb/EXANGULARJS/123"
{"NN":"Bantel"}
>>> curl "localhost/ydb/EXANGULARJS/124"
{}
>>>
```

Single-Page-Apps with AngularJS – Screenshot



localhost:8080/ydb_fcgi/EXANGULARJS.html

ID:	1
Vorname:	Winfried
Nachname:	Bantel

{"ERROR":0,"ERRTXT":"OK","ID-WRITTEN":"1"}

[✓] Logs nicht leeren [ ] Cache deaktivieren Keine Drosselung HAR

Status	Methode	Host	D...	P...	Urspr...	Typ	Übertragen	Größe	
304	GET	localhost:80...	EX...	HTTP/1.1	document	html	Aus Cache	778 B	18 ms
304	GET	ajax.googlea...	an...	HTTP/2.0	script	js	Aus Cache	0 B	16 ms
304	GET	localhost:80...	EX...	HTTP/1.1	script	js	Aus Cache	568 B	14 ms
404	GET	localhost:80...	fav...	HTTP/1.1	img	html	Aus Cache	169 B	
200	PUT	localhost:80...	1	HTTP/1.1	xhr	json	245 B	43 B	

JSON
- NN: Bantel
- VN: Winfried

5 Anfragen | 1,52 KB / 245 B übertragen | Beendet: 15,82 s | DOMContentLoaded: 200 ms | load: 311 ms

HTML5-Server-Sent Events (SSE)

Advantages of SSE

- Browser can be informed about Server-Events
- No Polling (AJAX) required

Disadvantage of SSE

- Direction only server $\rightarrow$ browser

HTML5-SSE - HTML

```
<!DOCTYPE html>
<html><head><title>SSE with YottaDB</title>
<script>
function init() {
      var source = new EventSource("/ydb/fcgi-sse");
      source.addEventListener('message',f, false);
}
function f(event) {
      document.getElementById("data").firstChild.nodeValue =
      event.data;
}
</script>
</head>
<body onload="init()">
Data: <div id="data">???</div>
</body>
</html>
```

HTML5-SSE - M

- Only possible with direct-output
- See EXOUTPUT6-example
- Attention, the YottaDB-redirect-uri has to be the same as exsse.html line-nr 5!

```
EXSSE ; SSE-Schnittstelle
        s %fcgi("o","header","Content-Type")="text/event-stream;chars
        d HEADEROUT^FCGI
        f i=1:1:1000000 d
        . s txt2send=i_": "_$H
        . d DATAOUT^FCGI("data: "_txt2send_$C(13,10,13,10))
        . h 1
        s %fcgi("o","noout")=1
        q
```

HTTP-Authorization

- Example uses Basic-scheme
- Should only be used via HTTPS!
- In this example credentials are hard coded: User „W“ and passwort „B“

```
EXAUTHBASIC       ;
      s up=$G(%fcgi("i","header","HTTP_AUTHORIZATION"))
      s up=$$BASE64DECODE^FCGI($P(up," ",2))
      i up'="W:B" d  q  ; User must be W, password B
      . s %fcgi("o","header","status")="401 Unauthorized"
      . s %fcgi("o","header","WWW-Authenticate")="Basic realm=""ydb
      . w "<html><head><head><body>401 Unauthorized</body></html>"

      w "<html><head><head><body>This is secret</body></html>"
      q
```

```
>>> curl "localhost/ydb/exauthbasic"
<html><head><head><body>401 Unauthorized</body></html>
>>> curl "localhost/ydb/exauthbasic" -u "W:B"
<html><head><head><body>This is secret</body></html>
>>>
```