

Multimedia Engineering II

08 Datenhaltung: mongoDB NoSQL

Johannes Konert



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences



Agenda

- **Wiederholung: Vertiefung *SQL in node.js, Absicherung, Skalierung**
- **NoSQL und Performance**
 - **Konzeptvergleich mit mySQL**
 - **Alternativen**
- **mongoDB**
 - **Eigenschaften**
 - **Installation**
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- **_____**
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**



Bildquelle:
http://static.zehn.de/image/3a2e5985e6bcc6c726abe9b2f9a1a335/1248105898.4.081fdc6b871a6e5f1134554af404e040.Sahnehaeubchen_Fotolia_4150869_askaja.jpgthumb_187x141

Zusammenfassende Fragen und Wiederholung



Aufgabe:

- 1.** Sie schreiben die zusammenfassenden Fragen + Antworten selbst auf
- 2.** Nutzen Sie dazu die Moderationskarten
 - eine Seite Frage
 - Andere Seite Antwort(en)
+ ggf. Foliennummer v. heute
- 3.** Von jedem am Ende mindestens eine Karte bei mir abgeben

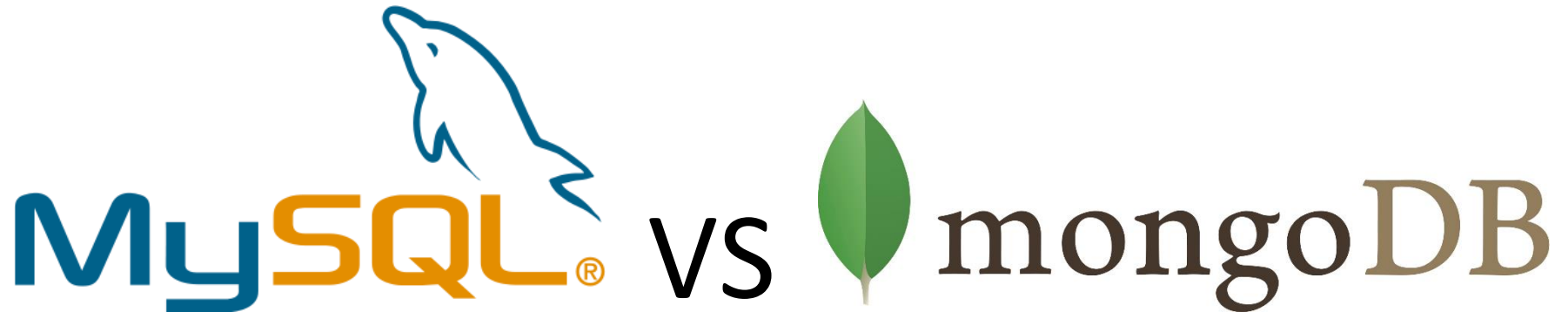


Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften
 - Installation
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- _____
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

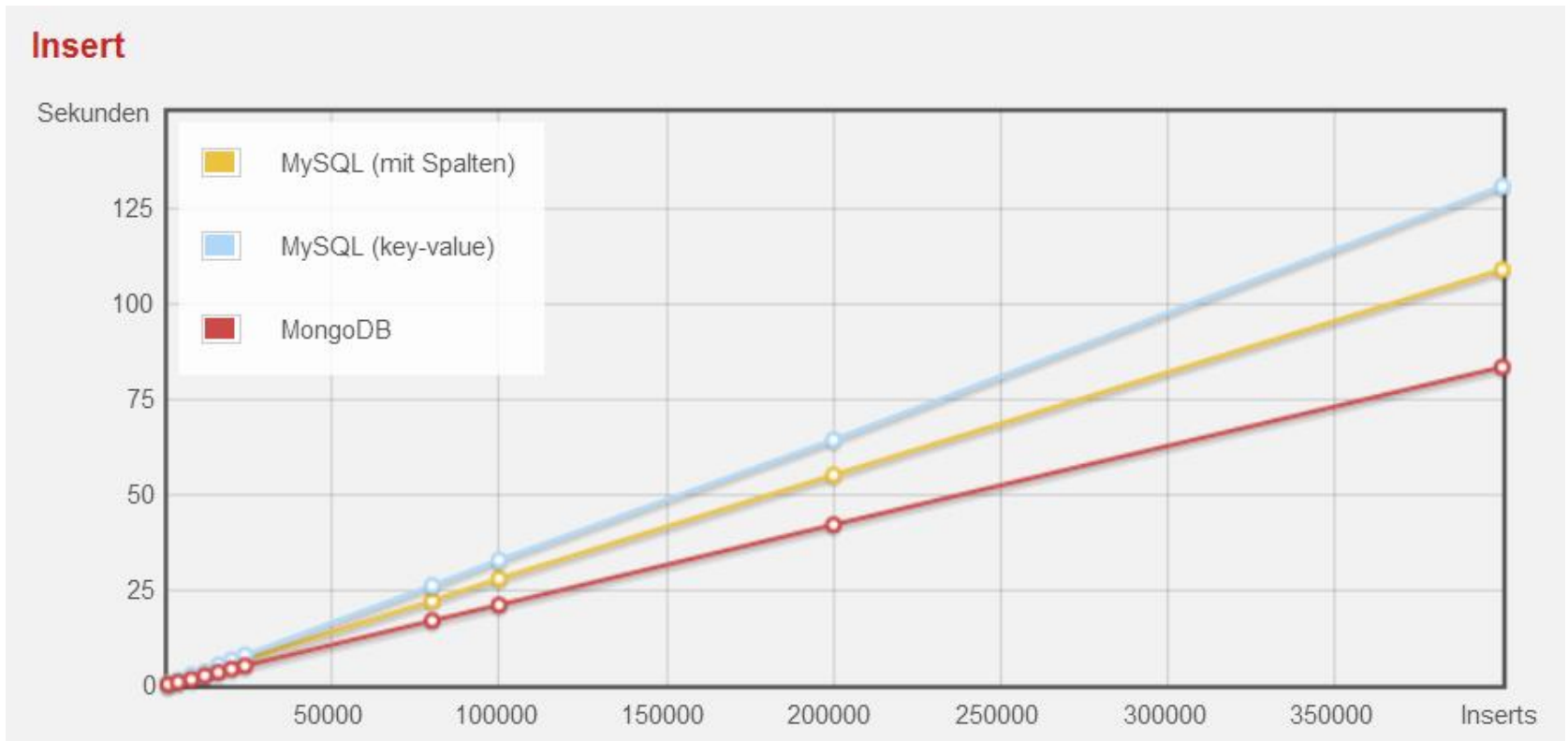
Was heißt NoSQL?

Not only SQL



Performance

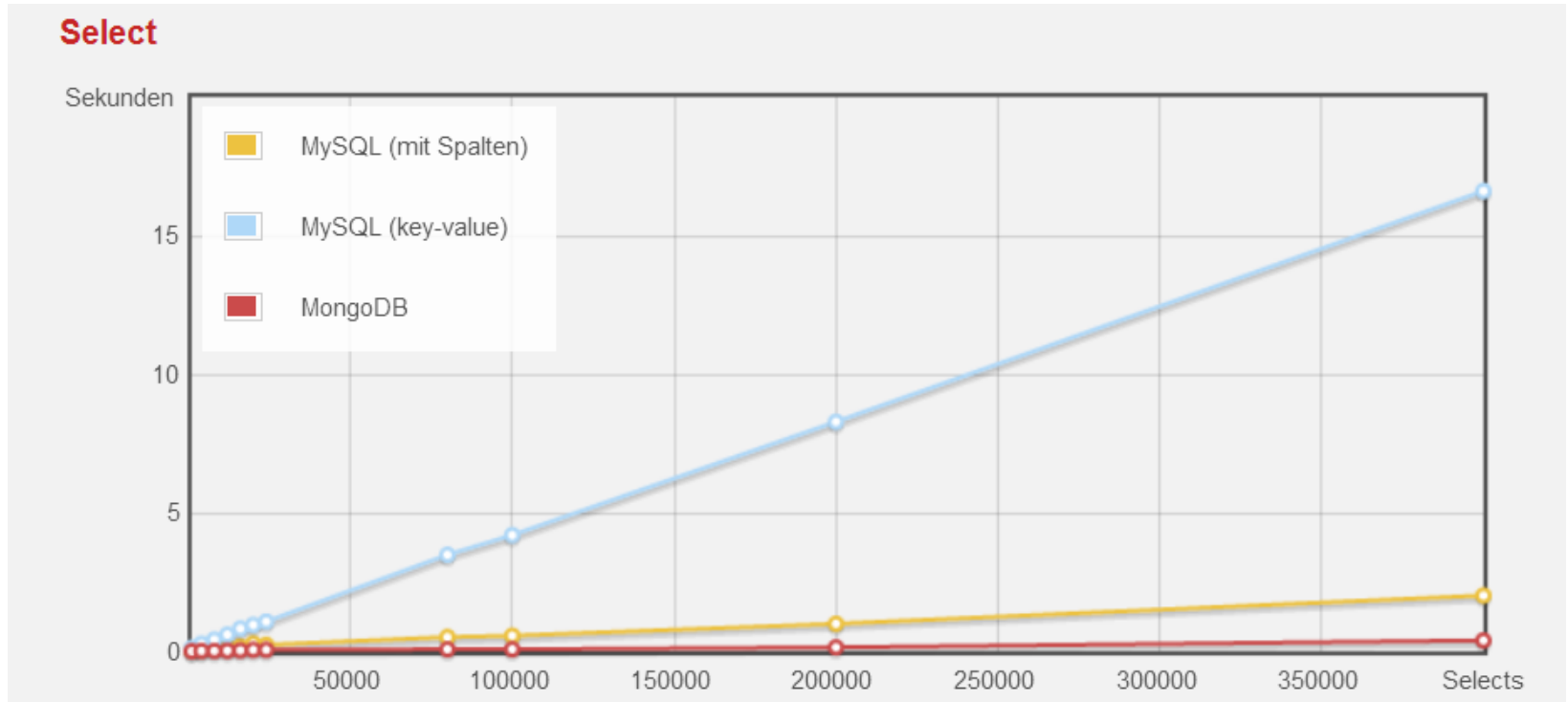
NoSQL Performance



MongoDB 30,6% schneller als MySQL

<http://skowron.biz/artikel/mysql-mariadb-vs-mongodb/>

NoSQL Performance

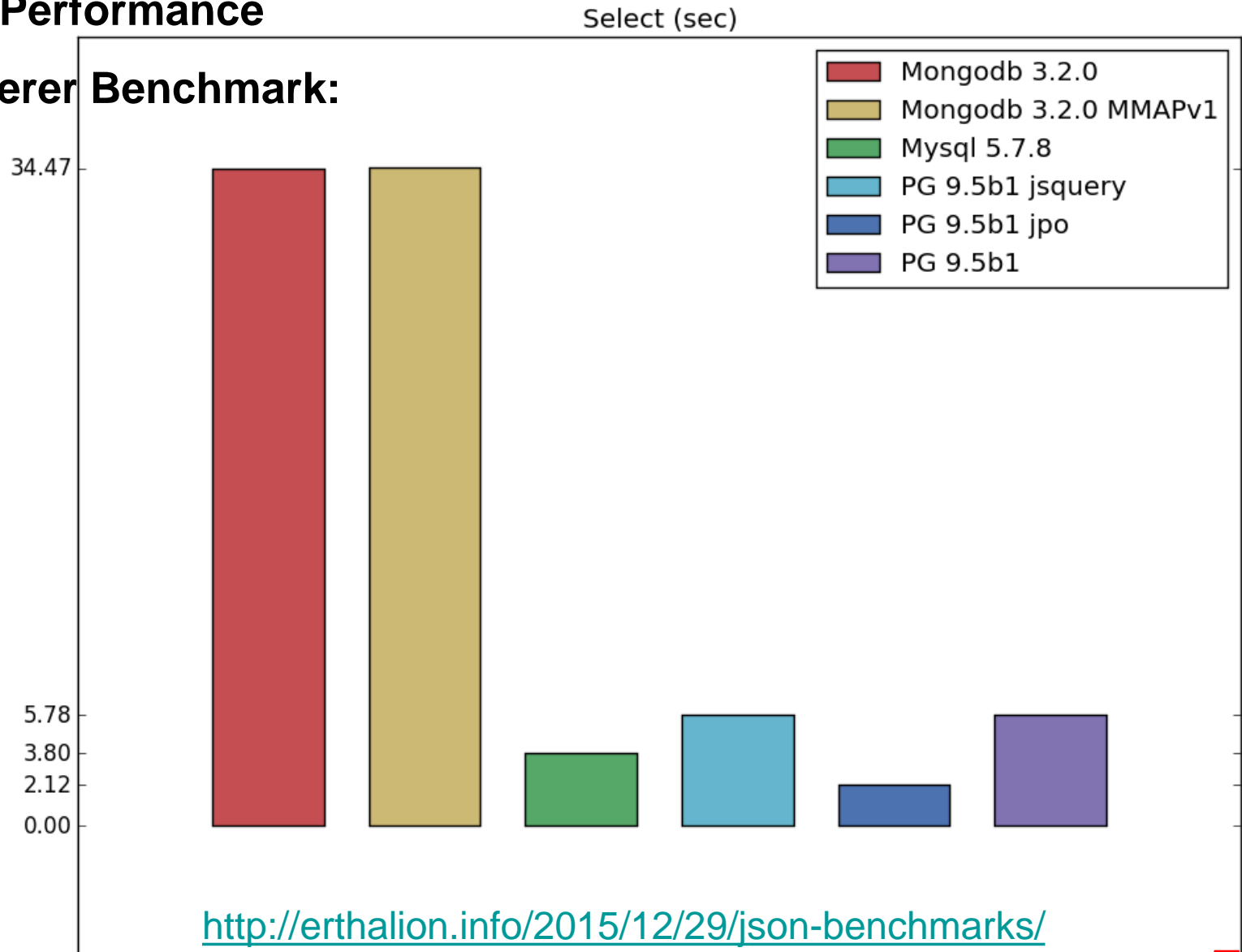


MongoDB 500% schneller als MySQL

<http://skowron.biz/artikel/mysql-mariadb-vs-mongodb/>

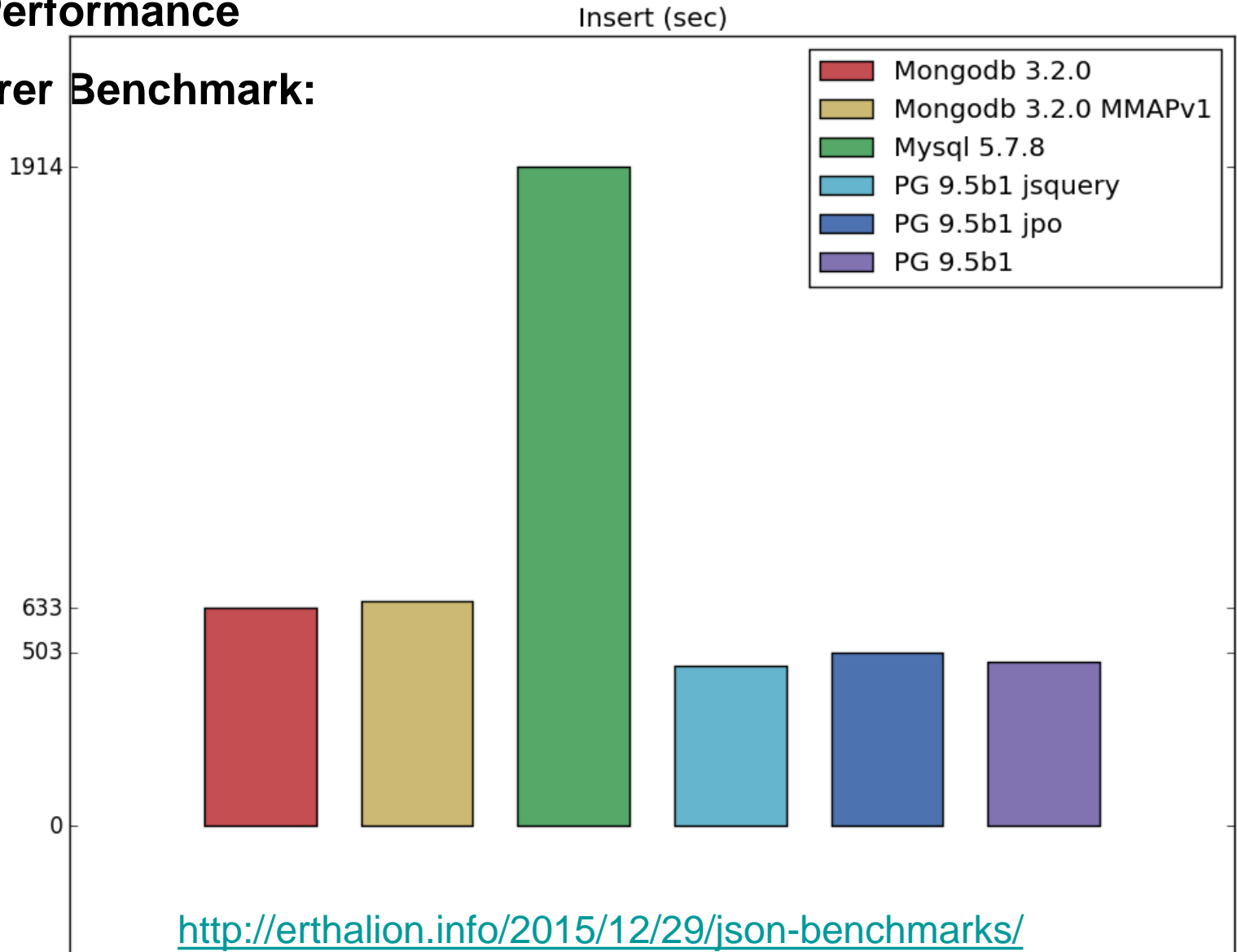
NoSQL Performance

Ein anderer Benchmark:



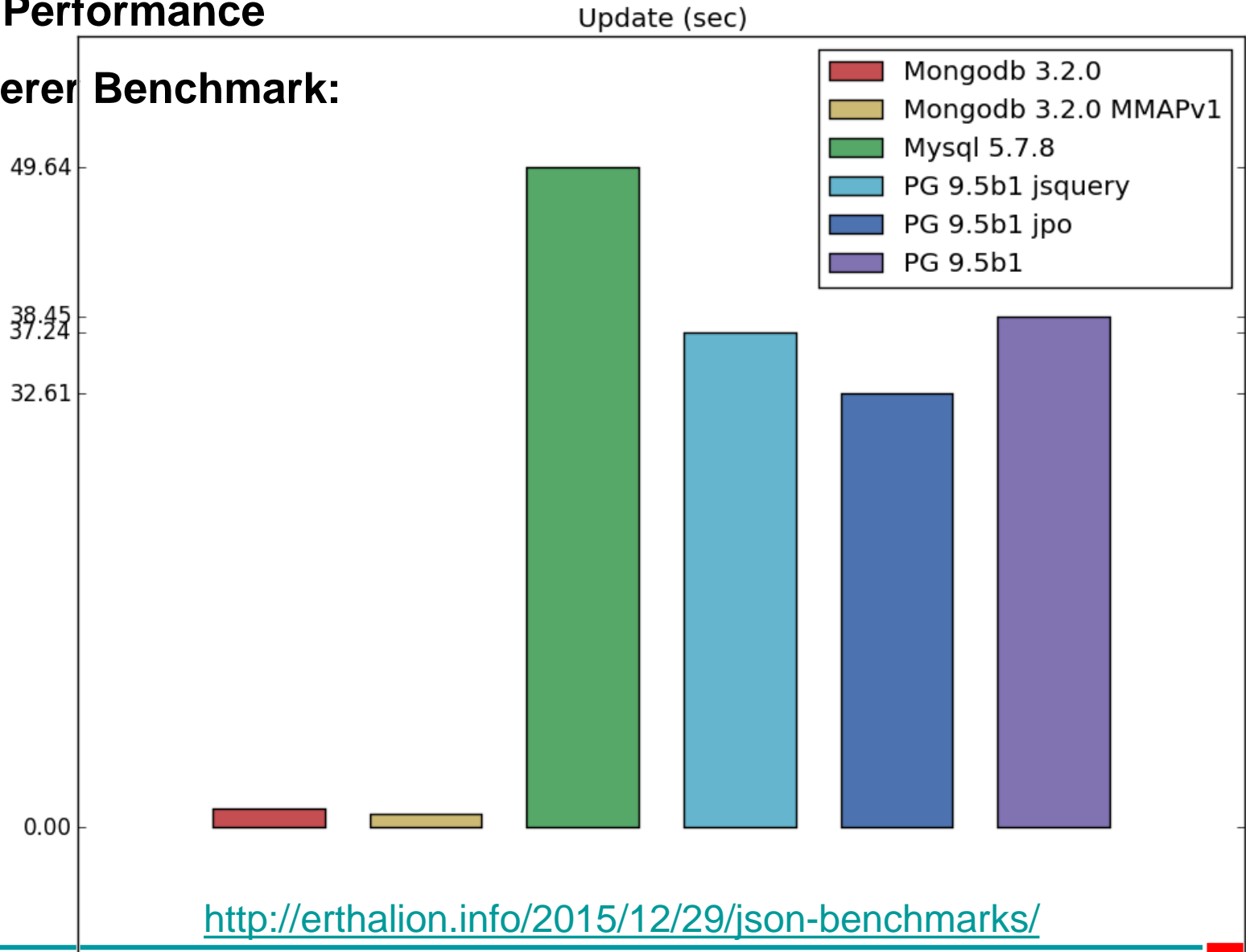
NoSQL Performance

Ein anderer Benchmark:



NoSQL Performance

Ein anderer Benchmark:



NoSQL Performance

Zwischenfazit

- **Es kommt auf den Test an**
(welche Konfiguration? Welche einfachen/Komplexen Anfragen?)
- **PostgreSQL hat inzwischen auch schnelle JSON und key-value Datenbank-Formate**
- **MongoDB punktet dann, wenn**
 - Einfache Inserts/Updates
 - Keine JOINS etc (keine Tabellenübergreifende Anfragen)

Agenda

- **Wiederholung**

- **NoSQL und Performance**

 - Konzeptvergleich mit MySQL**

 - **Alternativen**

- **mongoDB**

 - **Eigenschaften**

 - **Installation**

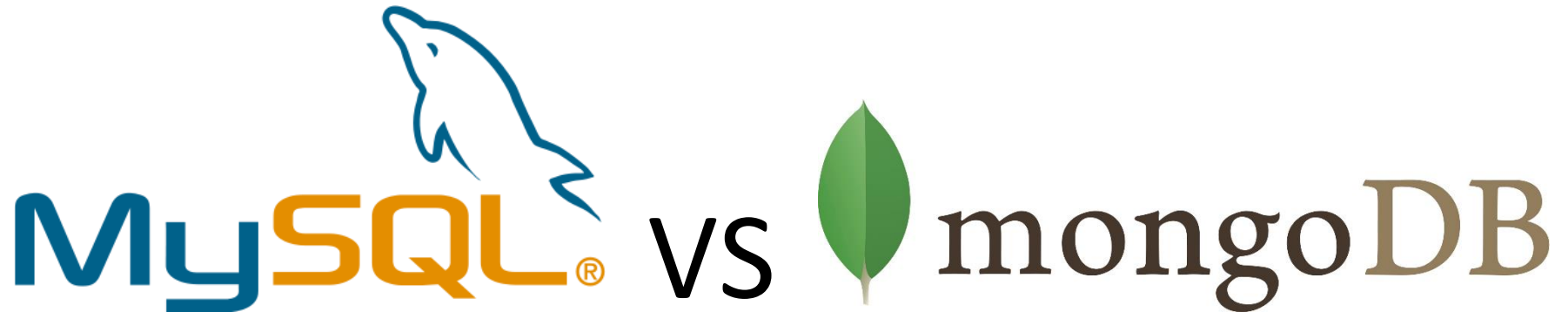
- **Mongojs + Aufgabe**

- **Mongoose + Aufgabe**



- **Zuordnungsaufgabe als Zusammenfassung**

- **Ausblick**



Konzeptvergleich

Konzeptvergleich

id	vorname	nachname	plz
0	Max	Mustermann	10133
1	Susi	Sonnenschein	10625
2	Bernd	Brot	10533

key	value
0	{ _id: 0, vorname: Max, nachname: Mustermann, plz: 10133 }
1	{ _id: 1, vorname: Susi, nachname: Sonnenschein, plz: 10625 }
2	{ _id: 2, vorname: Bernd, nachname: Brot, plz: 10533 }



SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents and linking
primary key Specify any unique column or column combination as primary key.	primary key In MongoDB, the primary key is automatically set to the _id field.
aggregation (e.g. group by)	aggregation pipeline See the SQL to Aggregation Mapping Chart .

<http://docs.mongodb.org/manual/reference/sql-comparison/>

SQL Schema Statements

```
CREATE TABLE users ( id MEDIUMINT NOT  
NULL AUTO_INCREMENT, user_id Varchar(30),  
age Number, status char(1), PRIMARY KEY (id) )
```

```
ALTER TABLE users ADD join_date DATETIME
```

MongoDB Schema Statements

Implicitly created on first [insert\(\)](#) operation. The primary key `_id` is automatically added if `_id` field is not specified.

```
db.users.insert( { user_id: "abc123", age: 55,  
status: "A" } )
```

However, you can also explicitly create a collection:

```
db.createCollection("users")
```

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document level, [update\(\)](#) operations can add fields to existing documents using the [\\$set](#) operator.

```
db.users.update( { }, { $set: { join_date: new  
Date() } }, { multi: true } )
```

<http://docs.mongodb.org/manual/reference/sql-comparison/>

Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

However, at the document

level, [update\(\)](#) operations can remove fields from documents using the [\\$unset](#) operator.

```
db.users.update( { },  
                { $unset: { join_date: "" } },  
                { multi: true } )
```

ALTER TABLE users **DROP COLUMN** join_date

CREATE INDEX idx_user_id_asc **ON**
users(user_id)

```
db.users.ensureIndex( { user_id: 1 } )
```

CREATE INDEX idx_user_id_asc_age_desc **ON**
users(user_id, age **DESC**)

```
db.users.ensureIndex( { user_id: 1, age: -1 } )
```

DROP TABLE users

```
db.users.drop()
```

<http://docs.mongodb.org/manual/reference/sql-comparison/>

SQL INSERT Statements

```
INSERT INTO users(user_id, age, status)  
VALUES ("bcd001", 45, "A")
```

MongoDB insert() Statements

```
db.users.insert( { user_id: "bcd001", age: 45,  
status: "A" } )
```

SQL Update Statements

```
UPDATE users SET status = "C" WHERE age >  
25
```

MongoDB update() Statements

```
db.users.update( { age: { $gt: 25 } },  
{ $set: { status: "C" } },  
{ multi: true } )
```

```
UPDATE users SET age = age + 3 WHERE  
status = "A"
```

```
db.users.update( { status: "A" },  
{ $inc: { age: 3 } },  
{ multi: true } )
```

<http://docs.mongodb.org/manual/reference/sql-comparison/>

SQL Delete Statements

DELETE FROM users **WHERE** status = "D"

DELETE FROM users

MongoDB remove() Statements

db.users.remove({ status: "D" })

db.users.remove()

<http://docs.mongodb.org/manual/reference/sql-comparison/>

SQL SELECT Statements

SELECT * **FROM** users

SELECT id, user_id, status **FROM** users

SELECT user_id, status **FROM** users

SELECT * **FROM** users **WHERE** status = "A"

SELECT user_id, status **FROM** users **WHERE** status = "A"

SELECT * **FROM** users **WHERE** status != "A"

SELECT * **FROM** users **WHERE** status = "A" **AND** age = 50

SELECT * **FROM** users **WHERE** status = "A" **OR** age = 50

MongoDB find() Statements

db.users.find()

db.users.find({ }, { user_id: 1, status: 1 })

db.users.find({ }, { user_id: 1, status: 1, _id: 0 })

db.users.find({ status: "A" })

db.users.find({ status: "A" },
 { user_id: 1, status: 1, _id: 0 })

db.users.find({ status: { \$ne: "A" } })

db.users.find({ status: "A", age: 50 })

db.users.find({ \$or: [{ status: "A" }, { age: 50 }] })

Viele weitere Beispiele siehe MongoDB-Referenz

<http://docs.mongodb.org/manual/reference/sql-comparison/>

Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - **Konzeptvergleich mit mySQL**

Alternativen

- **mongoDB**
 - **Eigenschaften**
 - **Installation**
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- _____
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

- **Alternativen**
 - **Apache Cassandra**
 - **CouchDB**

- **Reine Key-Value-Datenbanken**
 - **Google BigTable**
 - **Berkeley DB**

- **Objekt-Datenbanken**
 - **DB4O**

- **Key-Value-Erweiterungen und JSON-Erweiterungen bei**
 - **MySQL**
 - **PostgreSQL, ..**

Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen

- **mongoDB**

- **Eigenschaften**
- **Installation**

- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**



mongoDB



- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

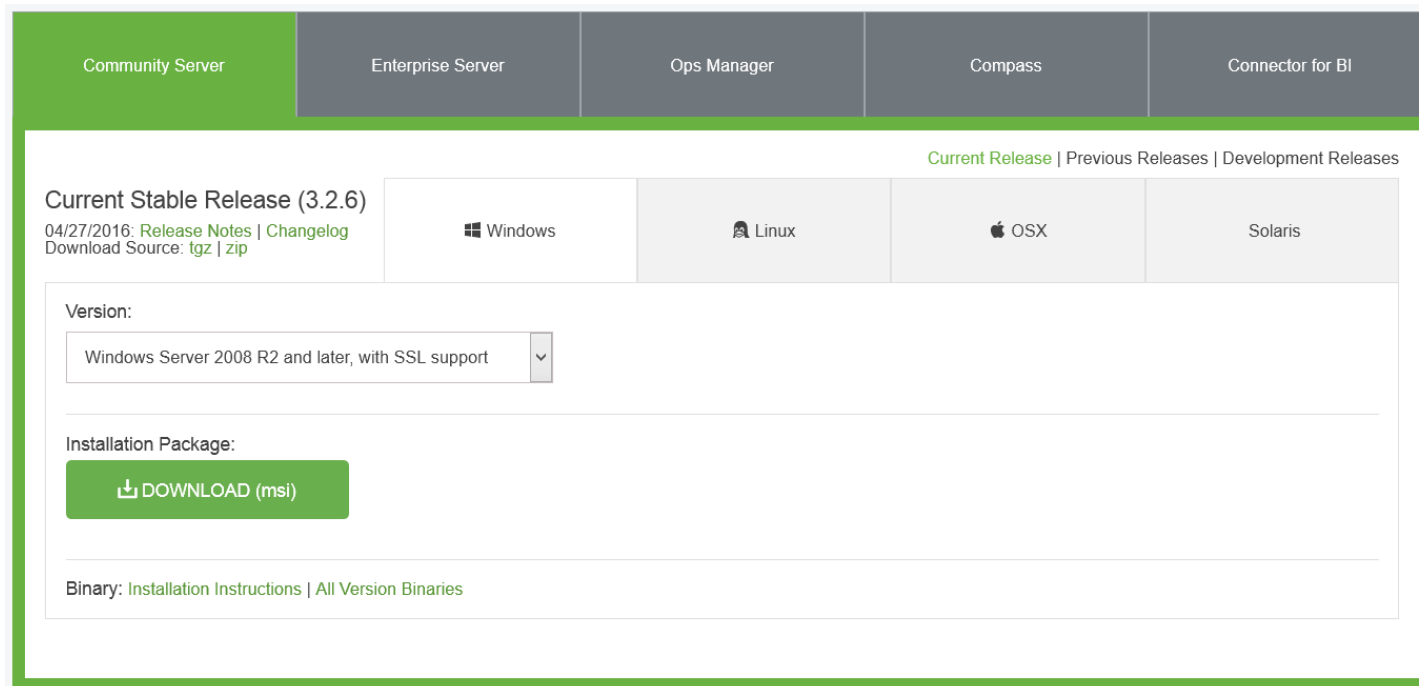
mongoDB

- Kommt von „humongous“ *
- führende NoSQL-Datenbank
- Open Source
- Dokumentenorientiert (JSON)
- Skalierbar (horizontal und vertikal)
- Syntax ist JavaScript-kompatibel!



mongoDB - Installation

- 1. Install (bspw. Windows)
 - <https://www.mongodb.com/download-center#community>



The screenshot shows the MongoDB download center interface. At the top, there are tabs for 'Community Server' (selected), 'Enterprise Server', 'Ops Manager', 'Compass', and 'Connector for BI'. Below the tabs, there are links for 'Current Release', 'Previous Releases', and 'Development Releases'. The 'Current Stable Release (3.2.6)' is highlighted, with a date of '04/27/2016' and links for 'Release Notes', 'Changelog', 'Download Source: tgz', and 'zip'. Below this, there are buttons for different operating systems: 'Windows' (selected), 'Linux', 'OSX', and 'Solaris'. Under the 'Windows' button, there is a 'Version:' dropdown menu showing 'Windows Server 2008 R2 and later, with SSL support'. Below the dropdown, there is an 'Installation Package:' section with a green 'DOWNLOAD (msi)' button. At the bottom, there is a 'Binary:' section with links for 'Installation Instructions' and 'All Version Binaries'.

<https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-windows/>

mongoDB - Installation

- 2. Datenbankordner erstellen

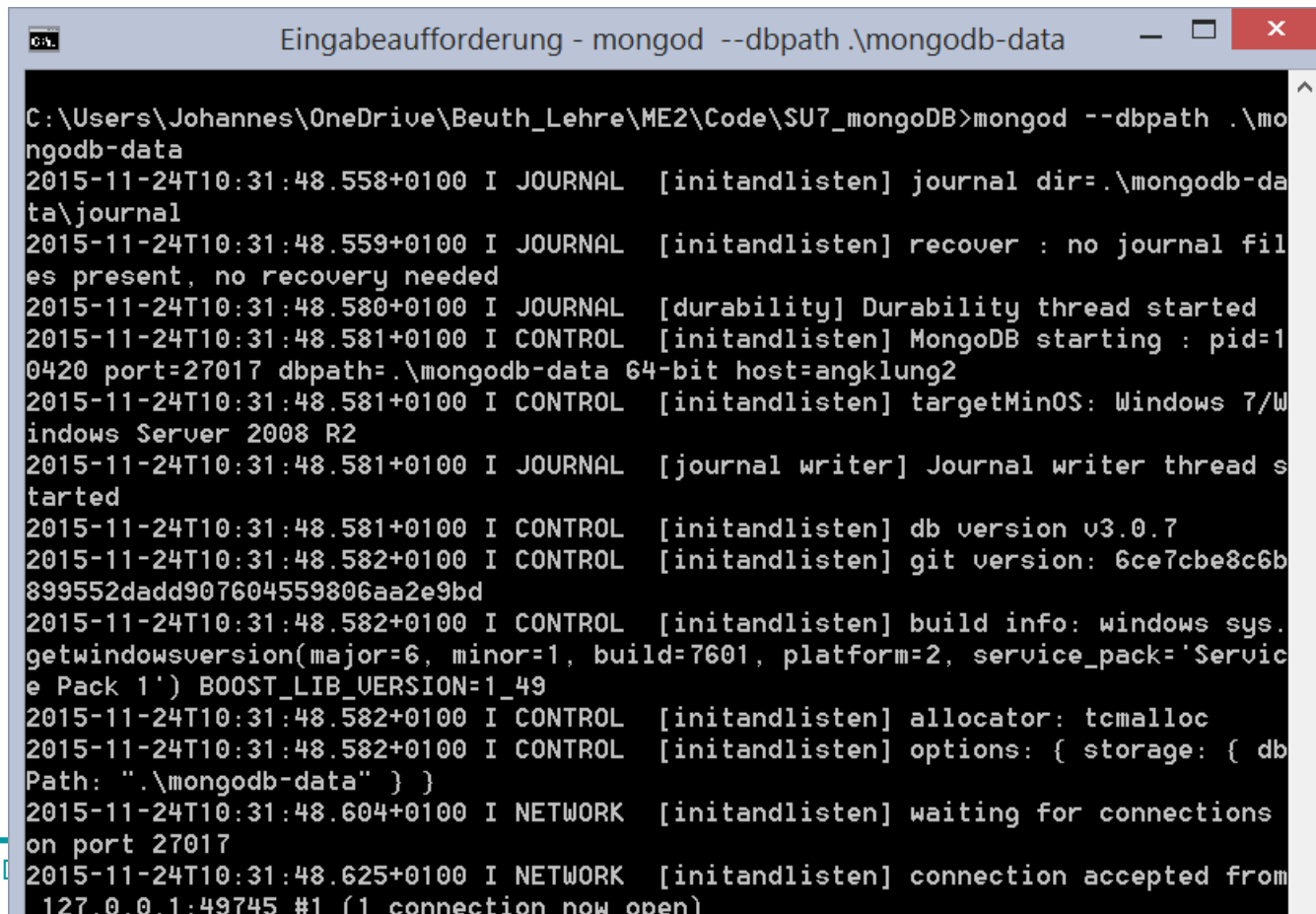
- data

z.B.:

- .\mongodb-data

mongoDB - Installation

- 3. mongoDB starten (Terminal bleibt offen)
 - `mongod.exe --dbpath .\mongodb-data`



```
Eingabeaufforderung - mongod --dbpath .\mongodb-data
C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU7_mongoDB>mongod --dbpath .\mo
ngodb-data
2015-11-24T10:31:48.558+0100 I JOURNAL [initandlisten] journal dir=.\mongodb-da
ta\journal
2015-11-24T10:31:48.559+0100 I JOURNAL [initandlisten] recover : no journal fil
es present, no recovery needed
2015-11-24T10:31:48.580+0100 I JOURNAL [durability] Durability thread started
2015-11-24T10:31:48.581+0100 I CONTROL [initandlisten] MongoDB starting : pid=1
0420 port=27017 dbpath=.\mongodb-data 64-bit host=angklung2
2015-11-24T10:31:48.581+0100 I CONTROL [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-11-24T10:31:48.581+0100 I JOURNAL [journal writer] Journal writer thread s
tarted
2015-11-24T10:31:48.581+0100 I CONTROL [initandlisten] db version v3.0.7
2015-11-24T10:31:48.582+0100 I CONTROL [initandlisten] git version: 6ce7cbe8c6b
899552dadd907604559806aa2e9bd
2015-11-24T10:31:48.582+0100 I CONTROL [initandlisten] build info: windows sys.
getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Servic
e Pack 1') BOOST_LIB_VERSION=1_49
2015-11-24T10:31:48.582+0100 I CONTROL [initandlisten] allocator: tcmalloc
2015-11-24T10:31:48.582+0100 I CONTROL [initandlisten] options: { storage: { db
Path: \".\mongodb-data\" } }
2015-11-24T10:31:48.604+0100 I NETWORK [initandlisten] waiting for connections
on port 27017
2015-11-24T10:31:48.625+0100 I NETWORK [initandlisten] connection accepted from
127.0.0.1:49745 #1 (1 connection now open)
```

mongoDB - Test

- `mongo.exe` nutzen als Client

```
C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU7_mongoDB>mongo
MongoDB shell version: 3.0.7
connecting to: test
>
```

Auf dem Server: (altes Terminal)

```
Sun Nov 10 13:32:11.798 [initandlisten] connection accepted from 127.0.0.1:54186
#1 <1 connection now open>
```

mongoDB - Collection „Test“

```
C:\Users\Johannes\OneDrive\Beuth_Lehre\ME2\Code\SU7_mongoDB>mongo
MongoDB shell version: 3.0.7
connecting to: test
> db.test.insert( { "beuth-course": "me2" } )
WriteResult({ "nInserted" : 1 })
> db.test.find()
{ "_id" : ObjectId("56542f8f1a58a5bf5ef5732e"), "beuth-course" : "me2" }
>
```

Merke: bei mongodb

- Ein **Verzeichnis** ist zentraler Speicherort aller Datenbanken
- Pro **Datenbank** gibt es eine gleichnamige Datei
- **Dokumentensammlungen** (Collections) haben einen eindeutigen Namen in einer solchen Datenbank
- und **einzelne Dokumente** sind JSON-konforme Objekte in diesen Collections

■ **Installation für Linux**

Agenda

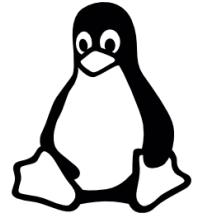
- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften

Installation

- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- _____
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**



mongoDB - Installation



- Linux – APT vorbereiten (Key für Package Signatur)
 - `sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927`
- Source-List erstellen
 - `echo "deb http://repo.mongodb.org/apt/debian wheezy/mongodb-org/3.2 main" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list`
- Installation
 - `sudo apt-get update`
 - `sudo apt-get install mongodb-org`

<https://docs.mongodb.com/manual/administration/install-on-linux/>

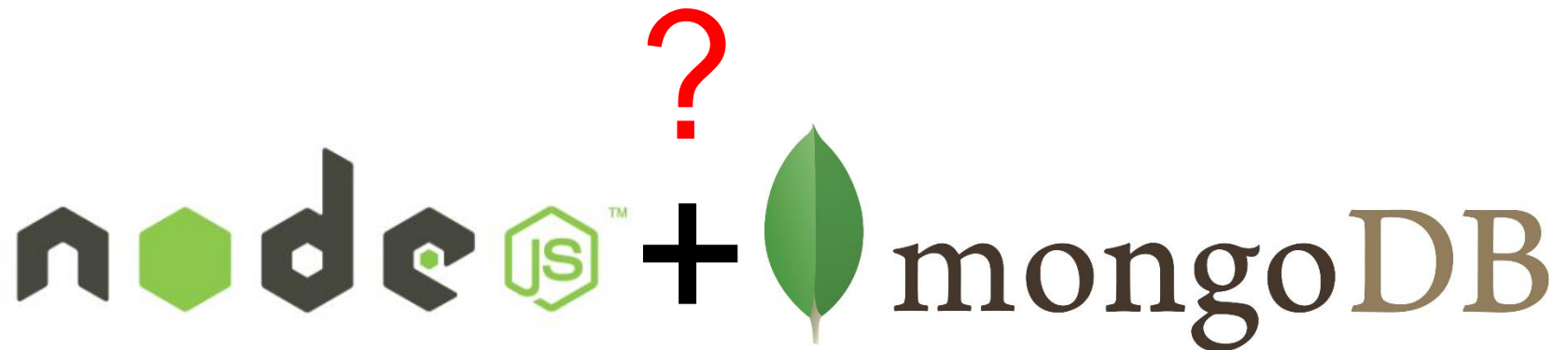
mongoDB - Installation

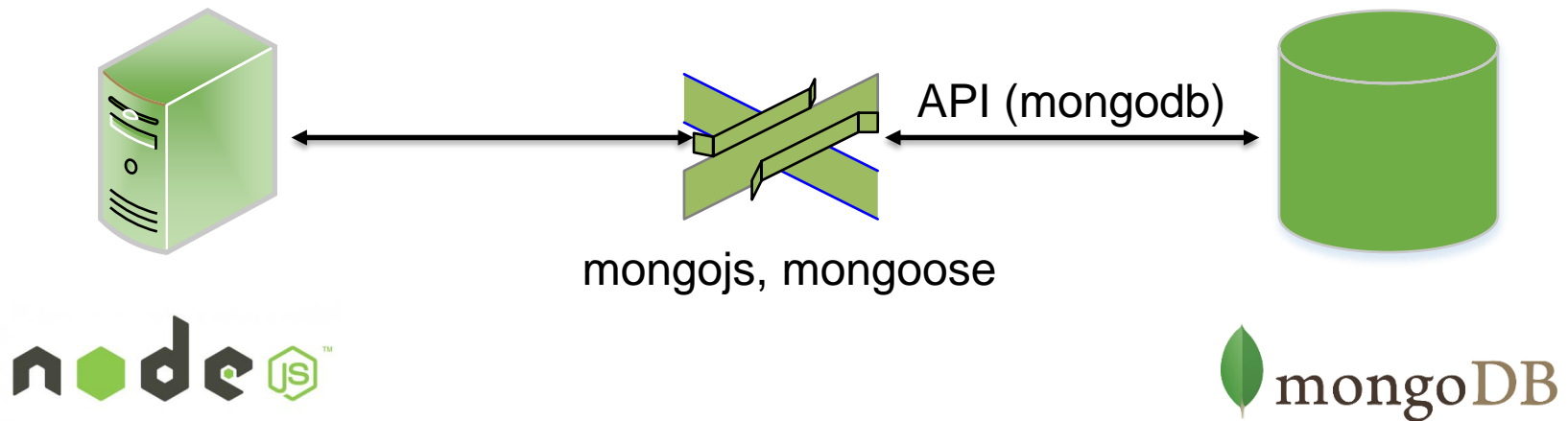
```
root@viwitra:~# apt-get install mongodb-org
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  mongodb-10gen
1 upgraded, 0 newly installed, 0 to remove and 80 not upgraded.
Need to get 87.9 MB of archives.
After this operation, 829 kB of additional disk space will be used.
Get:1 http://downloads-distro.mongodb.org/repo/ubuntu-upstart/ dist/10gen mongodb-10gen amd64 2.4.8 [87.9 MB]
Fetched 87.9 MB in 11s (7572 kB/s)
(Reading database ... 37554 files and directories currently installed.)
Preparing to replace mongodb-10gen 2.4.4 (using .../mongodb-10gen_2.4.8_amd64.deb) ...
arg: upgrade
mongodb stop/waiting
Unpacking replacement mongodb-10gen ...
Processing triggers for ureadahead ...
ureadahead will be reprofiled on next reboot
Setting up mongodb-10gen (2.4.8) ...
mongodb start/running, process 13357
root@viwitra:~# █
```

mongoDB - Test

■ Start über mongo

```
root@viwittra:~# mongo
MongoDB shell version: 2.4.8
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
    http://docs.mongodb.org/
Questions? Try the support group
    http://groups.google.com/group/mongodb-user
> db.test.save({a:1})
> db.test.find()
{ "_id" : ObjectId("527fa3b1b5fb8b2f03ff2e08"), "a" : 1 }
> █
```





- Für die eigentliche Verbindung wird ein Treiber benötigt. Nativer Treiber hierfür ist mongodb
 - <https://github.com/mongodb/node-mongodb-native>
- mongojs, mongoose sind Module, die eine Kommunikation zwischen nodeJS und mongoDB erleichtern.

mongojs vs. mongoose

mongojs

```
{
  "name": "mongojs",
  "version": "0.13.0",
  "repository": "git://github.com/mafintosh/mongojs.git",
  "author": "Mathias Buus Madsen <mathiasbuus@gmail.com>",
  "dependencies": {
    "thunky": "~0.1.0",
    "readable-stream": "~1.1.9",
    "mongodb": "1.4.0"
  },
  "scripts": {
    "test": "node ./tests"
  }
}
```

mongoose

```
{
  "name": "mongoose"
, "description": "Mongoose MongoDB ODM"
, "version": "3.8.12-pre"
, "author": "Guillermo Rauch <guillermo@learnboost.com>"
, "dependencies": {
    "mongodb": "1.4.5"
  , "hooks": "0.2.1"
  , "ms": "0.1.0"
  , "sliced": "0.0.5"
  , "muri": "0.3.1"
  , "mpromise": "0.4.3"
  , "mpath": "0.1.1"
  , "regexp-clone": "0.0.1"
  , "mquery" : "0.7.0"
  }
...
}
```


Kurzer Vergleich: mongojs und mongoose

■ **MongoJS**

- **Direkter Zugriff auf die Datenbank**
- **Flexibler Umgang mit Struktur**
- **Einfach in der Handhabung**
- **Native und einfache Mongo-Syntax**

■ **Mongoose**

- **Arbeit mit Schemata und Modellen**
- **Komplexe Anforderungen können umgesetzt werden (z.B. Referenzen aka. JOINS)**
- **Umständlich für minimale Anforderungen**

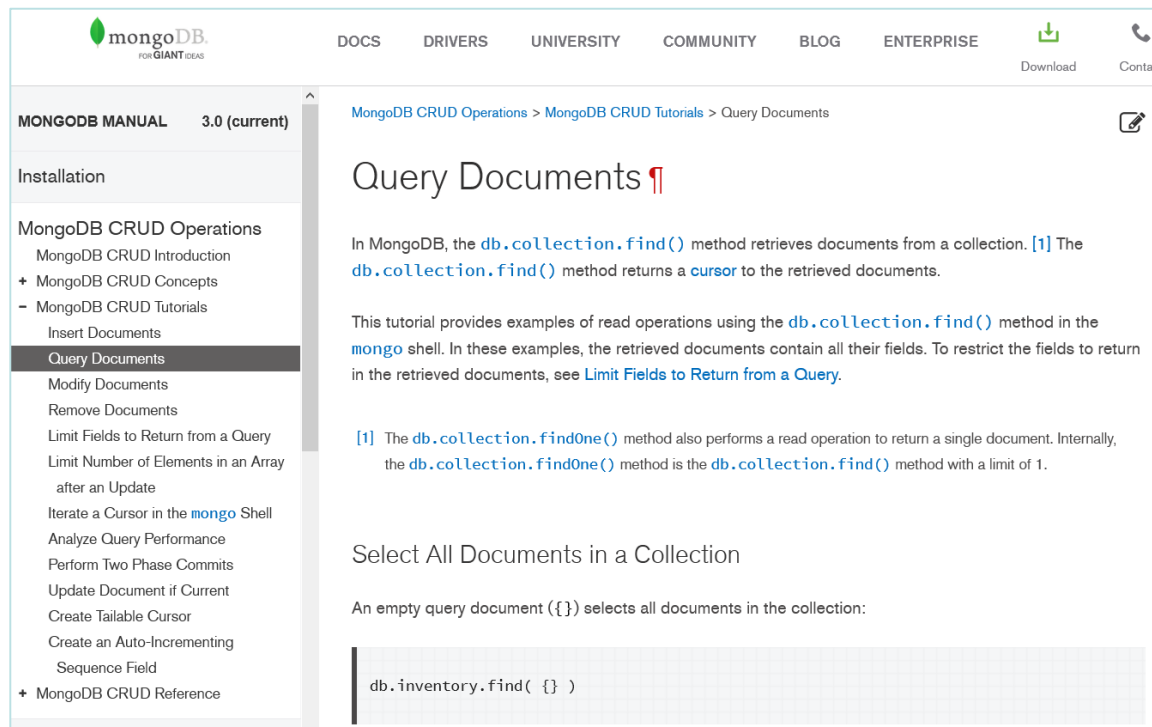
Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften
 - Installation
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- _____
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

node.js + mongoDB mit mongojs

Node Paket „mongojs“

- `npm install --save mongojs`
- `mongojs` emuliert die offizielle MongoDB API
 - Also Syntax wie im Terminal, aber direkt in `node.js`



The screenshot shows the MongoDB Manual website. The left sidebar contains a navigation menu with the following items: MONGODB MANUAL 3.0 (current), Installation, MongoDB CRUD Operations (with sub-items: MongoDB CRUD Introduction, MongoDB CRUD Concepts, MongoDB CRUD Tutorials, Insert Documents, Query Documents (highlighted), Modify Documents, Remove Documents, Limit Fields to Return from a Query, Limit Number of Elements in an Array after an Update, Iterate a Cursor in the mongo Shell, Analyze Query Performance, Perform Two Phase Commits, Update Document if Current, Create Tailable Cursor, Create an Auto-Incrementing Sequence Field, and MongoDB CRUD Reference), and MongoDB CRUD Reference. The main content area is titled 'Query Documents' and contains the following text: 'In MongoDB, the `db.collection.find()` method retrieves documents from a collection. [1] The `db.collection.find()` method returns a `cursor` to the retrieved documents. This tutorial provides examples of read operations using the `db.collection.find()` method in the `mongo` shell. In these examples, the retrieved documents contain all their fields. To restrict the fields to return in the retrieved documents, see [Limit Fields to Return from a Query](#). [1] The `db.collection.findOne()` method also performs a read operation to return a single document. Internally, the `db.collection.findOne()` method is the `db.collection.find()` method with a limit of 1. Select All Documents in a Collection An empty query document (`{}`) selects all documents in the collection:

```
db.inventory.find( {} )
```

Node Paket „mongojs“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

ALT

```
var store = require('./blackbox/store.js');
```

NEU

```
var mongojs = require('mongojs');  
var db = mongojs('mongodb://localhost:27017/me2',  
  ['tweets'],  
  {authMechanism: 'ScramSHA1'});
```

Standard-Port
der mongoDB

Gewünschte
Datenbank

Zu benutzende
Collections

Letzter Parameter: Objekt mit Optionen.
Bei MongoDB ^3.0 muss hier
ScramSHA1 genommen werden

Node Paket „mongojs“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

ALT

```
store.select('tweets')
```

Zu benutzende Collections sind dann als Objekt in db verfügbar

NEU

```
db.tweets.find({}, function(err, items) {  
  // check err, use items;  
});
```

Suchfilter-Objekt. Kann auch weggelassen werden.

Node Paket „mongojs“: Programmierbeispiel

■ Ziel: Ersetzen von store.js durch mongodb

ALT

```
var id = store.insert('tweets', req.body);  
  
store.replace('tweets', req.params.id, req.body);  
  
store.remove('tweets', req.params.id);
```

NEU

```
db.tweets.insert(req.body, function(err, item) {  
    id = item._id  
});  
  
db.tweets.findAndModify( {  
    query: { _id: mongojs.ObjectId(req.params.id) },  
    update: req.body,  
    new: true  
},  
function(err, item) {  
  
});  
  
db.tweets.remove({ _id: mongojs.ObjectId(req.params.id) },  
function(err, result) {  
  
});
```

Node Paket „mongojs“: Programmierbeispiel

- **Ziel: Ersetzen von store.js durch mongodb**

ALT

```
var id = store.insert('tweets', req.body);  
  
store.replace('tweets', req.params.id, req.body);  
  
store.remove('tweets', req.params.id);
```

```

NEU db.tweets.insert(req.body, function(err, result) {
    id = item._id
});

db.tweets.findAndModify( {
    query: { _id: mongojs.ObjectId(item._id) },
    update: req.body,
    new: true
},
function(err, item) {
    // console.log(item);
});

db.tweets.remove({ _id: mongojs.ObjectId(item._id) },
function(err, result) {
    // console.log(result);
});

```

Wie bei store.js besteht das Problem, dass jedes beliebige Objekt als Dokument eingefügt werden kann.

→ Konsistenzprüfung in der DB fehlt!

Node Paket „mongojs“

- Weitere Funktionen, siehe Doku
<https://github.com/mafintosh/mongojs>
 - Sortierung mit `.sort({name: 1})`
 - Kriterien mit `.find({level: {$gt: 90}})`
 - Limit, Offset mit `.limit(2).skip(1)` usw.
- Prinzipiell alle Funktionen, die mongoDB auch selbst hat, als JS-API

mongojs

A [node.js](#) module for mongodb, that emulates [the official mongodb API](#) as much as possible. It wraps [mongodb-core](#) and is available through [npm](#)

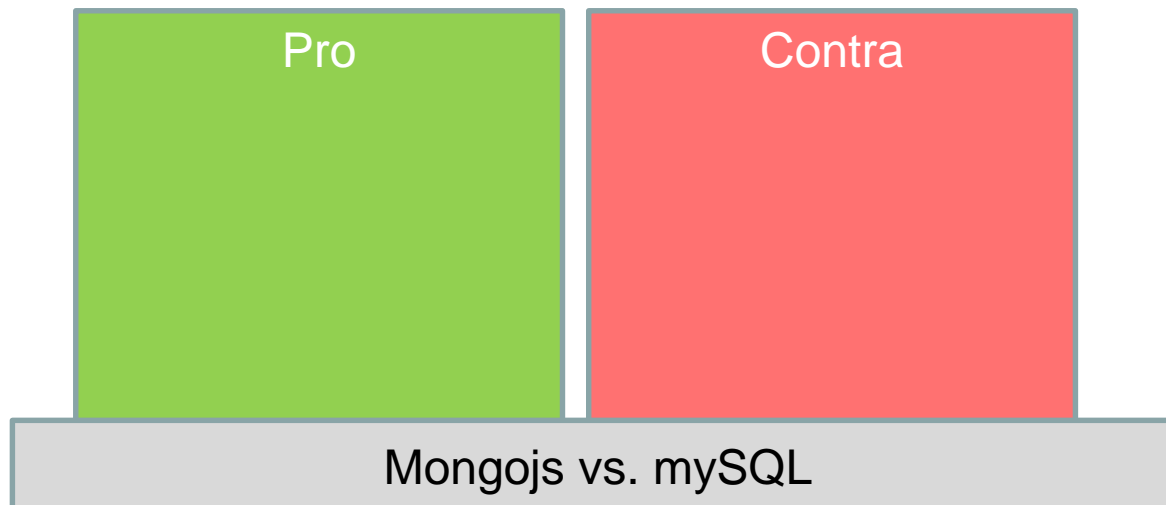
```
npm install mongojs
```

build passing code style standard

Zusammenfassung:

- **Aufgabe:** Welche Vor/Nachteile sehen Sie bei Nutzung von mongoDB + mongojs im Vergleich zu mySQL?

1. **Sammeln Sie pro Team entweder mind. zwei Vorteile oder mind. zwei Nachteile (2-3min Zeit)**
2. **Anschließend: Tafelsammlung**



Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften
 - Installation
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- _____
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

node.js + mongoDB mit mongoose

Node Paket „mongoose“

- `npm install --save mongoose`
 - Mongoose ist ein mongoDB Objekt-Mapper
 - Operationen
 - nicht direkt auf Collections via DB-Verbindung
 - Sondern: über Mongoose **Model** Instanzen



Was ist ein Model?

In mongoose:

- Eine Konstruktorfunktion, die Folgendes ermöglicht:
 - **Anlegen neuer Objekte basierend auf einem Schema**
 - **Zugriff auf die entsprechende DB-Collection des Models**

Code-Beispiel: Vergleich von mongojs und mongoose

■ Mongojs: direkter Abruf von der Collection der DB

```
var mongojs = require('mongojs');  
var db = mongojs('mydb', ['tweets']);  
db.tweets.find(function(err, docs) {  
    console.log(docs);  
});
```

■ Mongoose: ein Model wird erstellt (mittels eines Schemas) und darüber laufen die Anfragen

```
var mongoose = require('mongoose');  
mongoose.connect('mongodb://localhost/me2');  
  
var TweetModel = mongoose.model('Tweet', { message: String });  
  
TweetModel.find({}, function(err, tweets) {  
    console.log(tweets)  
});
```

```
var tweet = new TweetModel({ message: 'What a day..' });  
tweet.save(function(err) { });
```

Node Paket „mongoose“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

ALT

```
var store = require('./blackbox/store.js');
```

NEU

```
var mongoose = require('mongoose');  
var db = mongoose.connect('mongodb://localhost:27017/me2');  
  
var TweetModel = require('./models/tweets');
```

Node Paket „mongoose“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

NEU (in Datei ./models/tweets.js)

```
var mongoose = require('mongoose');  
var Schema = mongoose.Schema;
```

```
var TweetSchema = new Schema({  
  message: { type: String, required: true },  
  creator: { type: Schema.Types.ObjectId, ref: 'User' }  
}, {  
  timestamps: { createdAt: 'timestamp' }  
});
```

```
module.exports = mongoose.model('Tweet', TweetSchema);
```

Schema ist eine
Strukturdefinition (ähnlich einer
Tabellenstruktur bei MySQL)

Options-Objekt erlaubt u.a.
automatische Erstellung von
Timestamps

.model erstellt eine
Konstrukturfunktion, welche auch
statische Methoden für find(),
update usw. hat.

Node Paket „mongojs“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

ALT

```
store.select('tweets')
```

NEU

```
TweetModel.find({}, function(err, items) {  
    res.json(items);  
});
```

Node Paket „mongojs“: Programmierbeispiel

- Ziel: Ersetzen von store.js durch mongodb

ALT `var id = store.insert('tweets', req.body);`

NEU

```
var tweet = new TweetModel(req.body);
tweet.save(function(err) {
  if (!err) {
    res.status(201).json(tweet)
  }
  next(err);
});
```

Vorteile der Model-Nutzung

1. Erstellung mit new übernimmt nur Felder, die im Schema definiert sind und wandelt Typen um
2. .save(..) prüft automatisch auf Konsistenz und liefert Error err falls was fehlt/falsch ist

Node Paket „mongojs“: Programmierbeispiel

■ Ziel: Ersetzen von store.js durch mongodb

ALT

```
var id = store.insert('tweets', req.body);  
  
store.replace('tweets', req.params.id, req.body);  
  
store.remove('tweets', req.params.id);
```

NEU

```
var tweet = new TweetModel(req.body);  
tweet.save(function(err) {  
    if (!err) {  
        res.status(201).json(tweet)  
    }  
    next(err);  
});  
  
TweetModel.findByIdAndUpdate(req.params.id, req.body,  
    {new: true},  
    function(err, item) {  
  
    });  
  
TweetModel.findByIdAndRemove(req.params.id,  
    function(err, item) {  
  
    });
```

Mongoose-basierte API ansprechen mit Postman

http://localhost:300...



POST ▾

http://localhost:3000/tweets

Params

Authorization

Headers (2)

Body

Pre-request script



form-data



x-www-form-urlencoded



raw



binary

JSON (application/json)

1 ▾

{

2

| "message": "tweet without creator"

3

}

Body

Cookies

Headers (6)

Tests


Status 201 Created Time 55 ms

GET ▼ | http://localhost:3000/tweets | Params

Authorization | Headers (2) | Body | Pre-request script

No Auth ▼

Body | Cookies | Headers (6) | Tests | Status 200 OK

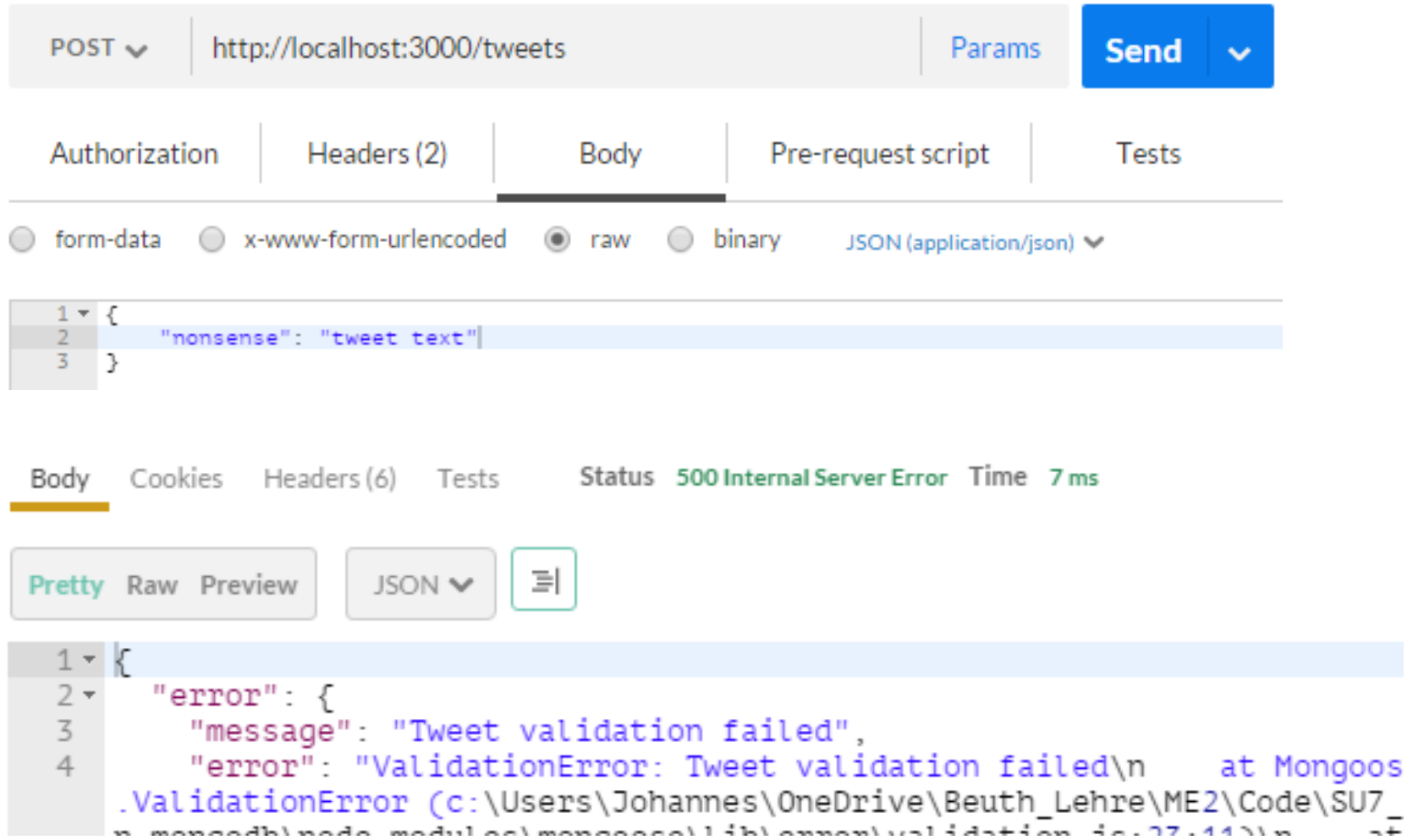
Pretty Raw Preview | JSON ▼ | 

```
1 [
2   {
3     "_id": "565454b28a5666ec3aba5862",
4     "updatedAt": "2015-11-24T12:14:42.000Z",
5     "timestamp": "2015-11-24T12:14:42.000Z",
6     "message": "tweet without creator",
7     "__v": 0
8   }
9 ]
```

mongoDB vergibt _id selbst

Version: wird hochgezählt bei Manipulation

Mongoose-basierte API ansprechen mit Postman



POST Params

Authorization Headers (2) **Body** Pre-request script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```
1 {  
2   "nonsense": "tweet text"  
3 }
```

Body Cookies Headers (6) Tests Status **500 Internal Server Error** Time 7 ms

Pretty Raw Preview JSON ▼


```
1 {  
2   "error": {  
3     "message": "Tweet validation failed",  
4     "error": "ValidationError: Tweet validation failed\n    at Mongoos  
.ValidationError (c:\\Users\\Johannes\\OneDrive\\Beuth_Lehre\\ME2\\Code\\SU7_  
n mongoose\\node_modules\\mongoose\\lib\\error\\validation.js:27:11)"
```

POST ▾

http://localhost:3000/tweets

Params

Send ▾

 ▾

Authorization

Headers (2)

Body

Pre-request script

Tests

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

JSON (application/json) ▾

```
1 {
2   "message": "tweet with creator",
3   "creator": "8923145ghfo71326045123"
4 }
```

Body

Cookies

Headers (6)

Tests

Status 500 Internal Server Error


Time 11 ms



Pretty

Raw

Preview

JSON ▾



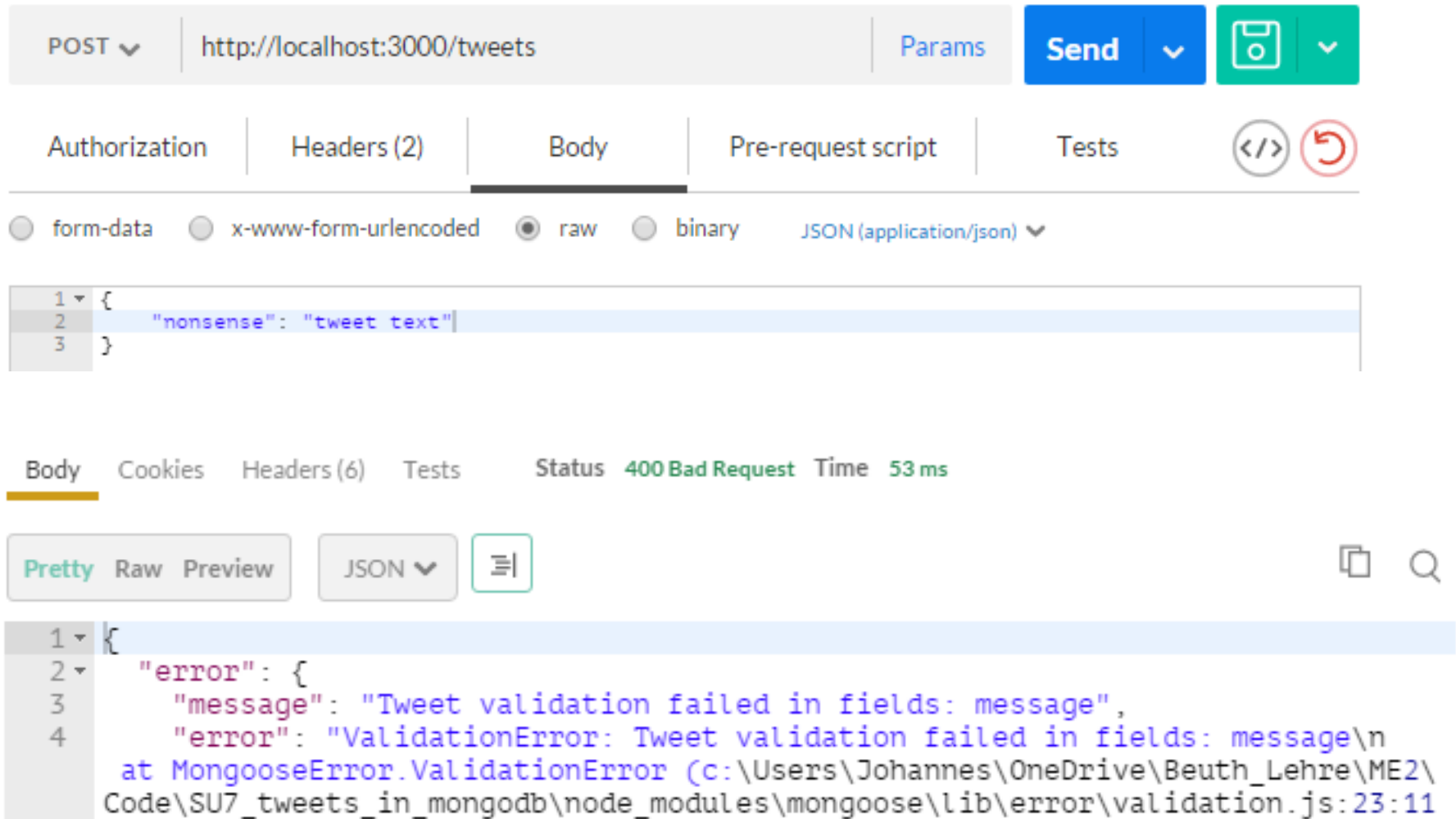
 

```
1 {
2   "error": {
3     "message": "Tweet validation failed",
4     "error": "ValidationError: Tweet validation failed\n    at MongooseError\n    .ValidationError (c:\\Users\\Johannes\\OneDrive\\Beuth_Lehre\\ME2\\Code\\SU7_tweets_i\nn_mongodb\\node_modules\\mongoose\\lib\\error\\validation.js:23:11)\n    at model\n    .Document.invalidate (c:\\Users\\Johannes\\OneDrive\\Beuth_Lehre\\ME2\\Code\\SU7_twee\n    ts_in_mongodb\\node_modules\\mongoose\\lib\\document.js:1280:32)\n    at model\n    Document.set (c:\\Users\\Johannes\\OneDrive\\Beuth_Lehre\\ME2\\Code\\SU7_tweets_in m
```

Mongoose Validation Errors: Example POST

```
var tweet = new TweetModel(req.body);
tweet.save(function(err) {
  if (!err) {
    res.status(201).json(tweet)
  } else {
    err.status = 400;
    err.message += ' in fields: '
                  + Object.getOwnPropertyNames(err.errors);
  }
  next(err);
});
```


Mongoose Validation Errors: Example POST



POST Params

Authorization Headers (2) **Body** Pre-request script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

```
1 {  
2   "nonsense": "tweet text"  
3 }
```

Body Cookies Headers (6) Tests Status **400 Bad Request** Time 53 ms

Pretty Raw Preview JSON

```
1 {  
2   "error": {  
3     "message": "Tweet validation failed in fields: message",  
4     "error": "ValidationError: Tweet validation failed in fields: message\nat MongooseError.ValidationError (c:\\Users\\Johannes\\OneDrive\\Beuth_Lehre\\ME2\\Code\\SU7_tweets_in_mongodb\\node_modules\\mongoose\\lib\\error\\validation.js:23:11"
```

Mehr Doku zu Validation Errors: <http://mongoosejs.com/docs/validation.html>

Mongoose: Filtern



Mongoose: Filtern

- Nur bestimmte Attribute zurückliefern

```
TweetModel.find({}, 'message' , function(err, items) {  
    res.json(items);  
});
```

```
[  
  {  
    "_id": "565454b28a5666ec3aba5862",  
    "message": "tweet without creator"  
  }  
]
```

- Auch möglich als Query via Fluent Interfaces zusammenzustellen

```
var query = TweetModel.find({  });  
query.select('message').exec(function(err, items) { ... });
```

Mongoose: Filtern

■ Weitere Filter

```
var query = TweetModel.find({ message: /first/ })  
  .where('likes').gt(0).lt(1000)  
  .limit(10)  
  .sort('-timestamp')  
  .select('message timestamp');  
query.exec(function(err, items) { });
```

■ Achtung: Bei sort() sollte ein index dafür angelegt worden sein!

```
TweetSchema.index({ timestamp: 1 });
```

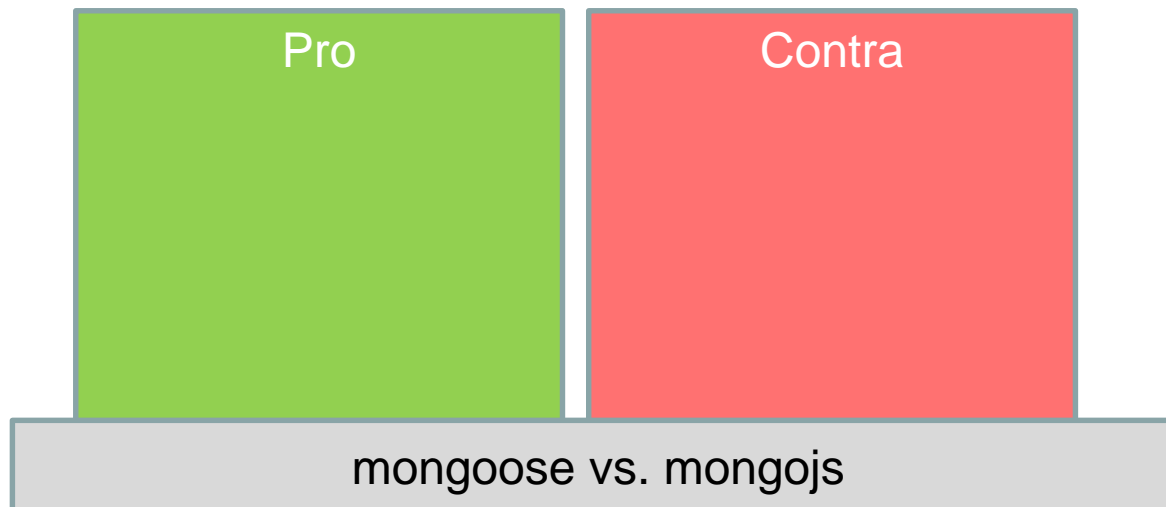
■ Siehe auch Doku

<http://mongoosejs.com/docs/queries.html>

Zusammenfassung:

- **Aufgabe:** Welche Vor/Nachteile sehen Sie bei Nutzung von mongoDB + mongoose **im Vergleich zu mongojs**?

1. **Sammeln Sie pro Team entweder mind. zwei Vorteile oder mind. zwei Nachteile (2-3min Zeit)**
2. **Anschließend: Tafelsammlung**



Quellen für APIs

- **mongoDB**
 - <https://github.com/mongodb/node-mongodb-native>
- **MongoJS**
 - <https://github.com/mafintosh/mongojs>
- **Mongoose**
 - <https://github.com/learnboost/mongoose>

Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften
 - Installation
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- **node-restful**
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**



REST APIs mit node-restful



- **npm install --save node-restful**
 - Eine Middleware
 - basierend auf Mongoose Model
 - Erstellt Handler für alle nötigen REST Methoden automatisch
 - Erstellt automatisch
 - GET /resources
 - GET /resources/:id
 - POST /resources
 - PUT /resources/:id
 - DELETE /resources/:id

REST APIs mit node-restful



- **npm install --save node-restful**

```
var restful = require('node-restful');  
var mongoose = require('mongoose');  
mongoose.connect('mongodb://localhost:27017/me2');  
  
var TweetSchema = require('./models/tweet-schema.js');  
  
var TweetModel = restful.model('Tweets', TweetSchema);  
TweetModel.methods(['get', 'post', 'put', 'delete']);  
TweetModel.register(app, '/tweets');  
  
logger('activated REST API for tweets');
```

- **Das ist alles.**

Mit register(..) hängt sich node-restful in app.use(..) entsprechend ein.

(Nutzung von MongoDB, mongoose oder node-restful für Übungsblatt 4 leider nicht erlaubt)

REST APIs mit node-restful

■ Zum Vergleich: Der Code vorher

```
app.get('/tweets', function(req,res,next) {
    TweetModel.find({}, 'message' , function(err,
        res.json(items);
    });
});

app.post('/tweets', function(req,res,next) {
    var tweet = new TweetModel(req.body);
    tweet.save(function(err) {
        if (!err) {
            res.status(201).json(tweet)
        } else {
            err.status = 400;
            err.message += ' in fields: ' + Object
        }
        next(err);
    });
});
...
```

```
app.get('/tweets/:id', function(req,res,next) {
    res.json(store.select('tweets', req.params.id));
});

app.delete('/tweets/:id', function(req,res,next) {
    TweetModel.findByIdAndRemove(req.params.id, function(err, it
        if (err || !item) {
            err = err || new Error("item not found");
            err.status = 404;
        } else {
            res.status(200).end();
        }
        next(err);
    });
});

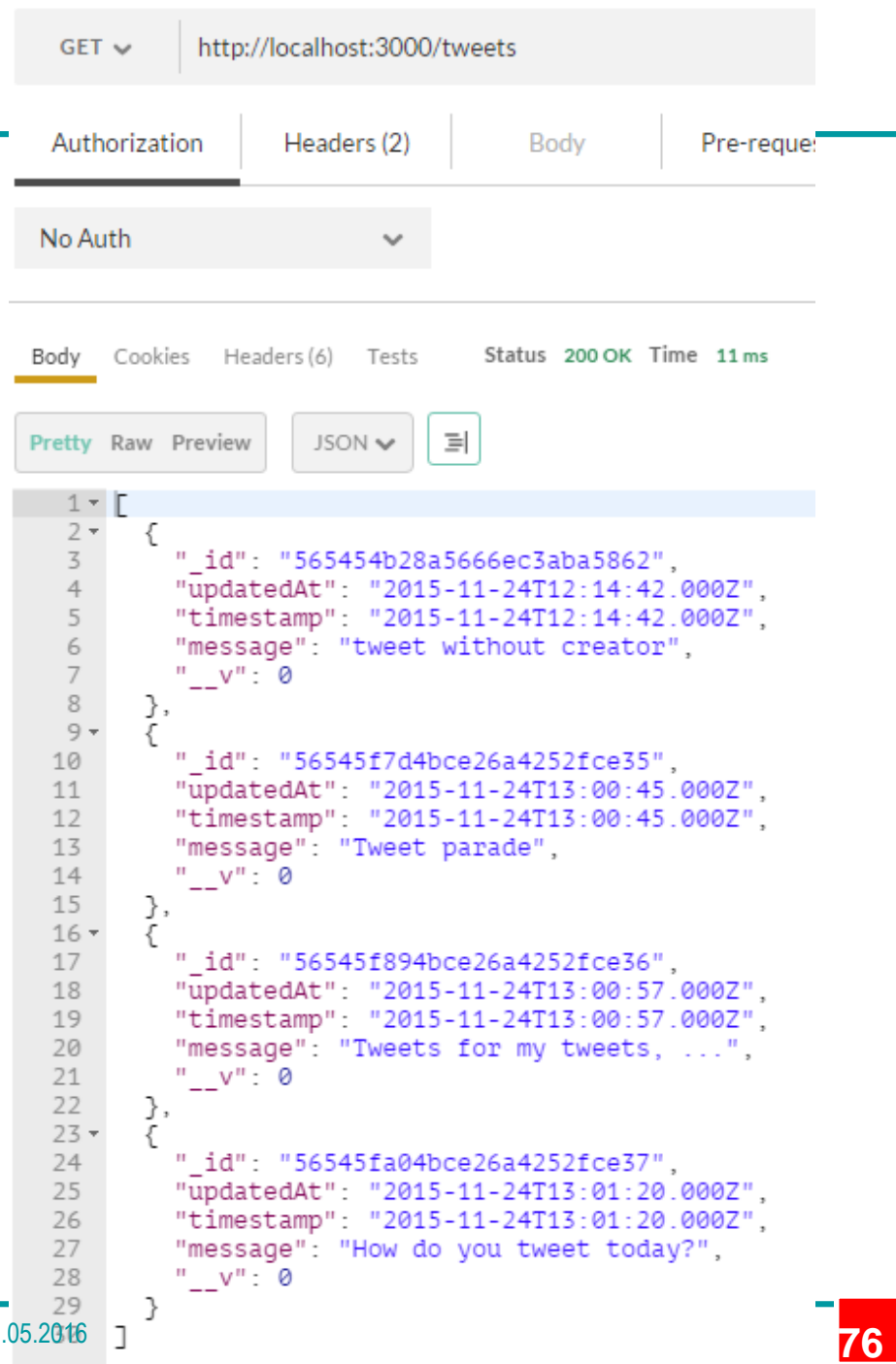
app.put('/tweets/:id', function(req,res,next) {
    TweetModel.findByIdAndUpdate(req.params.id, req.body, {new:
        if (err) {
            err.status = 400;
        }
        res.status(200).end();
        next(err);
    });
});
});
```

REST APIs mit node-restful

- **npm install --save node-restful**

- **Liefert Funktionen mit für**

- **Blättern**
- **Filtern**
- **Suche**



GET

Authorization Headers (2) Body Pre-reqs

No Auth

Body Cookies Headers (6) Tests Status 200 OK Time 11 ms

Pretty Raw Preview JSON

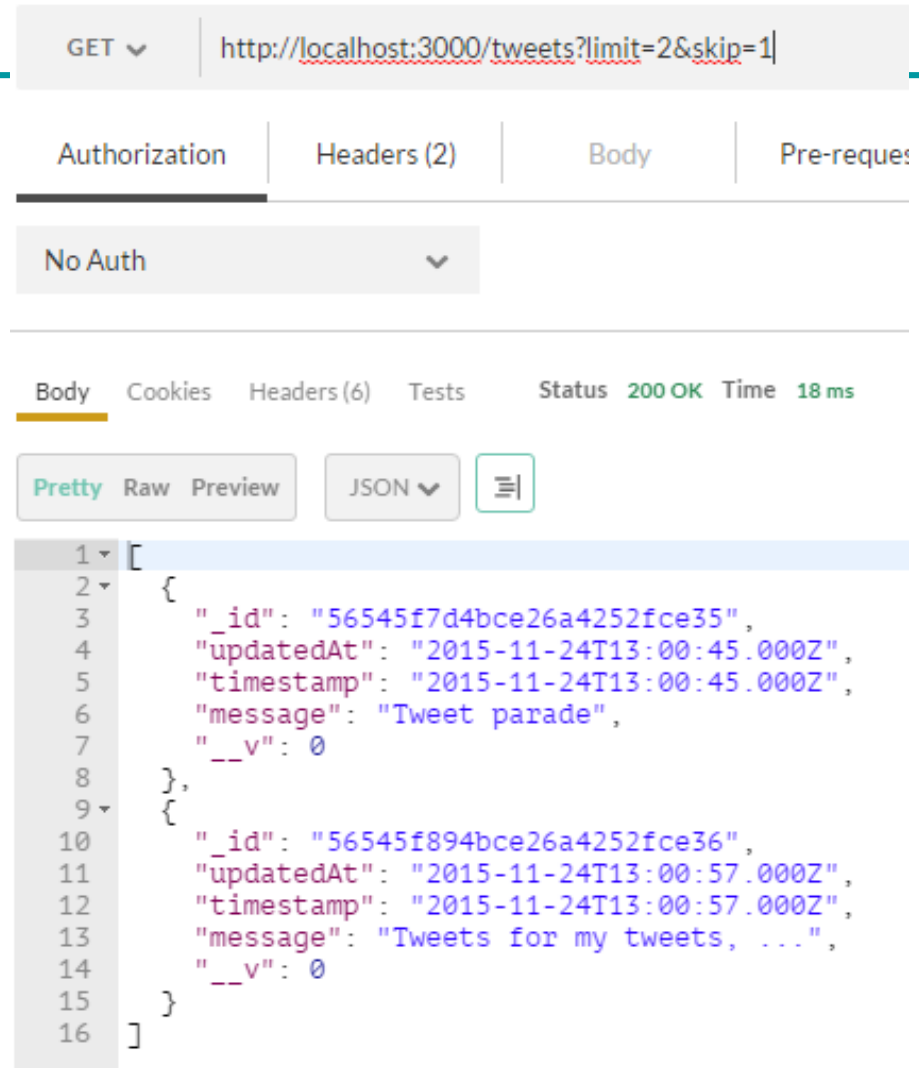
```

1 [
2   {
3     "_id": "565454b28a5666ec3aba5862",
4     "updatedAt": "2015-11-24T12:14:42.000Z",
5     "timestamp": "2015-11-24T12:14:42.000Z",
6     "message": "tweet without creator",
7     "__v": 0
8   },
9   {
10    "_id": "56545f7d4bce26a4252f3e35",
11    "updatedAt": "2015-11-24T13:00:45.000Z",
12    "timestamp": "2015-11-24T13:00:45.000Z",
13    "message": "Tweet parade",
14    "__v": 0
15  },
16  {
17    "_id": "56545f894bce26a4252f3e36",
18    "updatedAt": "2015-11-24T13:00:57.000Z",
19    "timestamp": "2015-11-24T13:00:57.000Z",
20    "message": "Tweets for my tweets, ...",
21    "__v": 0
22  },
23  {
24    "_id": "56545fa04bce26a4252f3e37",
25    "updatedAt": "2015-11-24T13:01:20.000Z",
26    "timestamp": "2015-11-24T13:01:20.000Z",
27    "message": "How do you tweet today?",
28    "__v": 0
29  }
30 ]

```

REST APIs mit node-restful

- `npm install --save node-restful`
- **Blättern:** `?limit=2&skip=1`
- **(liefert tweets 2 und 3)**



GET `http://localhost:3000/tweets?limit=2&skip=1`

Authorization | Headers (2) | Body | Pre-requests

No Auth

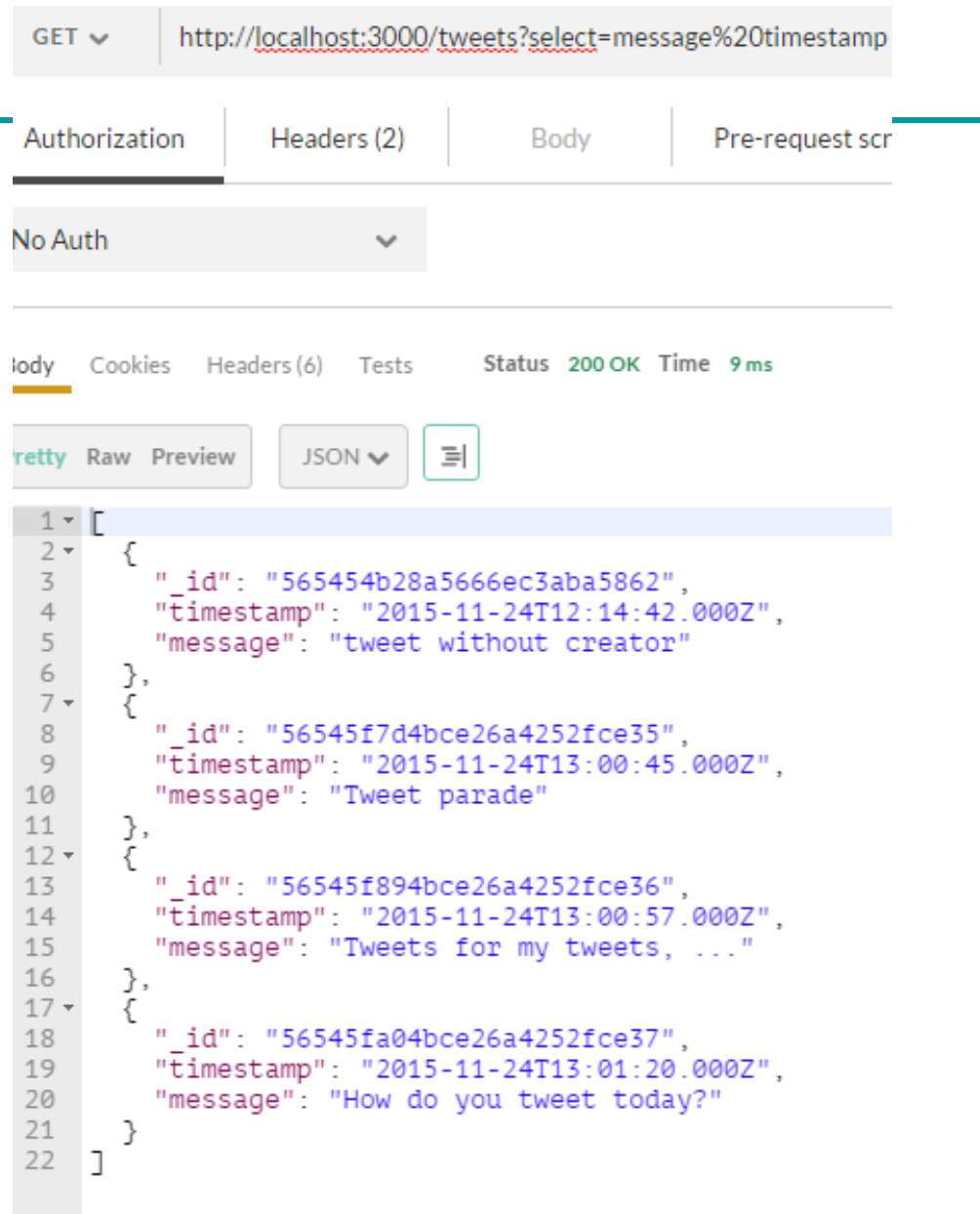
Body | Cookies | Headers (6) | Tests | Status **200 OK** Time **18 ms**

Pretty Raw Preview JSON

```
[
  {
    "_id": "56545f7d4bce26a4252fce35",
    "updatedAt": "2015-11-24T13:00:45.000Z",
    "timestamp": "2015-11-24T13:00:45.000Z",
    "message": "Tweet parade",
    "__v": 0
  },
  {
    "_id": "56545f894bce26a4252fce36",
    "updatedAt": "2015-11-24T13:00:57.000Z",
    "timestamp": "2015-11-24T13:00:57.000Z",
    "message": "Tweets for my tweets, ...",
    "__v": 0
  }
]
```

REST APIs mit node-restful

- **npm install --save node-restful**
- **Filtern:**
?select=message%20timestamp



GET <http://localhost:3000/tweets?select=message%20timestamp>

Authorization Headers (2) Body Pre-request script

No Auth

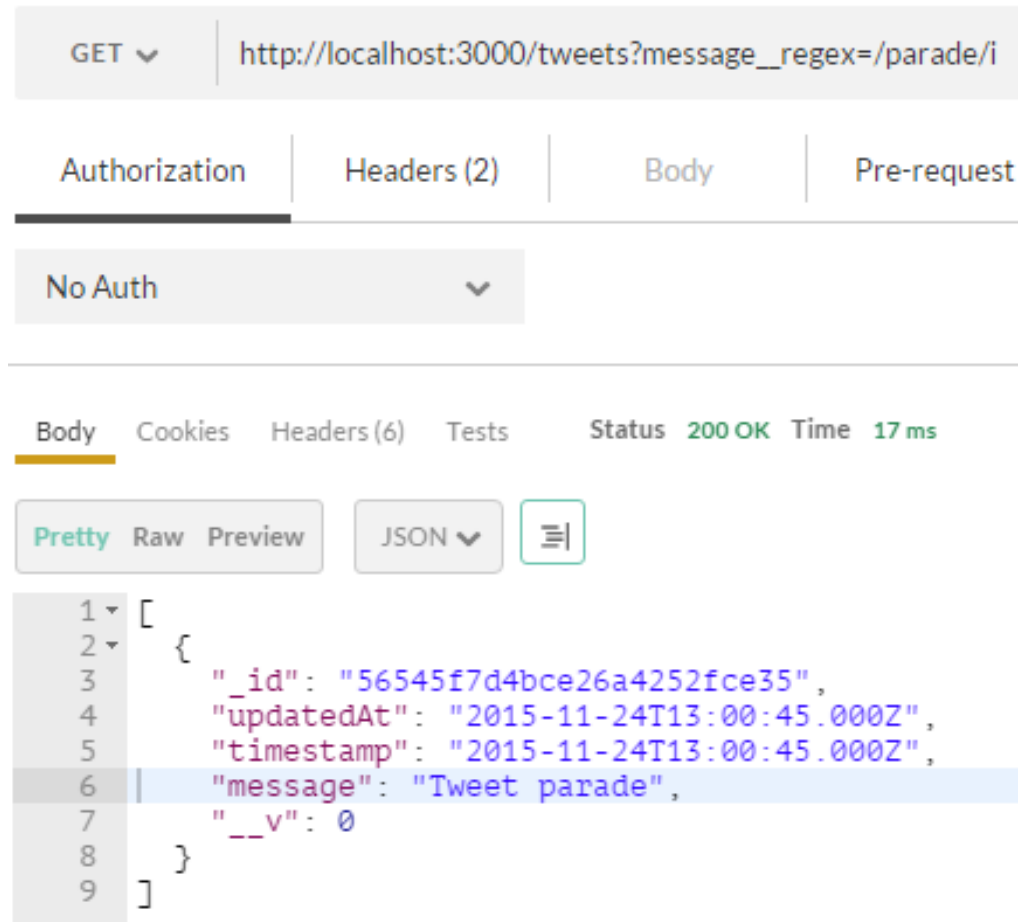
body Cookies Headers (6) Tests Status 200 OK Time 9 ms


pretty Raw Preview JSON

```
[
  {
    "_id": "565454b28a5666ec3aba5862",
    "timestamp": "2015-11-24T12:14:42.000Z",
    "message": "tweet without creator"
  },
  {
    "_id": "56545f7d4bce26a4252f3e35",
    "timestamp": "2015-11-24T13:00:45.000Z",
    "message": "Tweet parade"
  },
  {
    "_id": "56545f894bce26a4252f3e36",
    "timestamp": "2015-11-24T13:00:57.000Z",
    "message": "Tweets for my tweets, ..."
  },
  {
    "_id": "56545fa04bce26a4252f3e37",
    "timestamp": "2015-11-24T13:01:20.000Z",
    "message": "How do you tweet today?"
  }
]
```


REST APIs mit node-restful

- **npm install --save node-restful**
- **Suche:**
? message__regex=/parade/i
- **..und weiteres, siehe**
<https://github.com/baugarten/node-restful>





GET  | http://localhost:3000/tweets?message__regex=/parade/i

Authorization | Headers (2) | Body | Pre-request

No Auth 

Body | Cookies | Headers (6) | Tests | Status 200 OK Time 17 ms

Pretty Raw Preview | JSON  

```
1 [
2   {
3     "_id": "56545f7d4bce26a4252fce35",
4     "updatedAt": "2015-11-24T13:00:45.000Z",
5     "timestamp": "2015-11-24T13:00:45.000Z",
6     "message": "Tweet parade",
7     "__v": 0
8   }
9 ]
```

Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - **Konzeptvergleich mit MySQL**
 - **Alternativen**
- **mongoDB**
 - **Eigenschaften**
 - **Installation**
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- **node-restful**
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

Zusammenfassende Abschlussübung zu heutigen Themen

Was passt wozu? (3min)



- Ordnen Sie je genau 2 Elemente von rechts nach links zu
- Vergleichen Sie erst anschließend mit den Kommilitonen/innen

Technologie

- 1.) mongoDB
- 2.) mongojs
- 3.) Mongoose
- 4.) node-restful

Aussagen/Begriffe

- A.) Model
- B.) Collections aus JSON Dokumenten
- C.) mongoDB-API Syntax in node.js
- D.) Validierung
- E.) schneller als MySQL für einfache Daten
- F.) Basiert auf Mongoose
- G.) generiert REST-Schnittstelle für Schemata
- H.) schlanker direkter DB-Zugriff aus node.js

Zusammenfassende Abschlussübung zu heutigen Themen

Was passt wozu? (3min)



- Ordnen Sie je genau 2 Elemente von rechts nach links zu
- Vergleichen Sie erst anschließend mit den Kommilitonen/innen

Technologie

1.) mongoDB

2.) mongojs

3.) Mongoose

4.) node-restful

Aussagen/Begriffe

- (3) A.) Model
- (1) B.) Collections aus JSON Dokumenten
- (2) C.) mongoDB-API Syntax in node.js
- (3) D.) Validierung
- (1) E.) schneller als MySQL für einfache Daten
- (4) F.) Basiert auf Mongoose
- (4) G.) generiert REST-Schnittstelle für Schemata
- (2) H.) schlanker direkter DB-Zugriff aus node.js

Zusammenfassende Fragen

■ Themen heute

- NoSQL
- Performance von mongoDB
- Mongojs vs. Mongoose
- node-restful

■ Ihre Karten

(1) Einsammeln

(2) Drei Beispiele ziehen wir direkt
(und nutzen die Karten wieder
nächstes Mal)

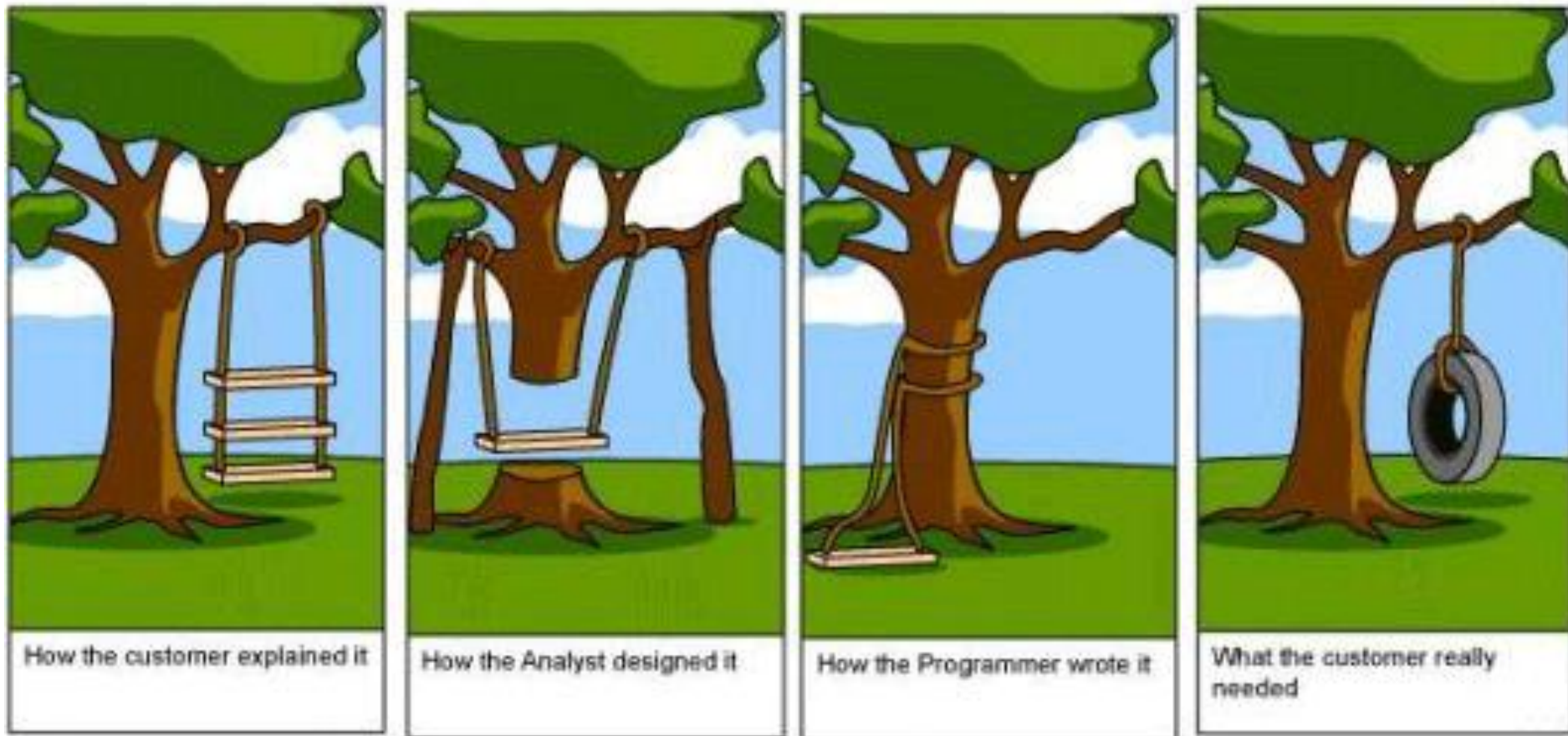


Agenda

- **Wiederholung**
- **NoSQL und Performance**
 - Konzeptvergleich mit **mySQL**
 - Alternativen
- **mongoDB**
 - Eigenschaften
 - Installation
- **Mongojs + Aufgabe**
- **Mongoose + Aufgabe**
- **node-restful**
- **Zuordnungsaufgabe als Zusammenfassung**
- **Ausblick**

Ausblick / Nächster Unterricht: REST APIs anzapfen mit Backbone.js

- (auch da gibt es Models)



Vielen Dank und bis zum nächsten Mal

Anhang

Exkurs: MEAN Stack im Einsatz. Ein winziges Beispiel

Being MEAN (server-side)

■ Starting a server and processing a request



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

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

app.listen(3000);
```

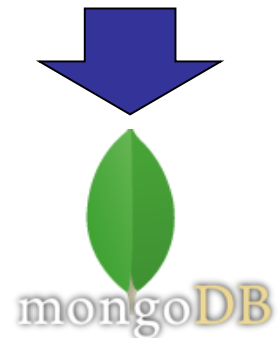
■ Using the database to save JSON*

```
var mongoDB = require('mongodb');

var dbClient = mongoDB.MongoClient;
dbClient.connect('mongodb://127.0.0.1:27017/mydb', function(err, db) {
  if(err) throw err;

  var tweet = {sender: 'John', text: 'Great lecture on MEAN stack'};
  db.collection('tweets').insert(tweet, function(){
    db.close();
  });
});
```

sender	John
text	Great lecture on MEAN stack



Being MEAN (server-side)

■ Together: using the database to send JSON



```
1 var express = require('express'), mongodb = require('mongodb');
2 var dbClient = mongodb.MongoClient;
3 var app = express();
4
5 app.get('/tweets/:name', function(req, res){
6   var name = req.params.name;
7
8   dbClient.connect('mongodb://127.0.0.1:27017/mydb', function(err, db) {
9     if(err) throw err;
10
11     db.collection('tweets').find({sender: name}).toArray(function(err, results) {
12       if(err) throw err;
13       res.send(results);
14       db.close();
15     });
16   });
17 });
```



sender	John	sender	John
text	Great lecture on MEAN stack	text	Yeah! My first tweet



Being MEAN (client-side)

■ Dynamic binding of view to model

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p>MyTweetApp</p>
5 <div ng-app="">
6
7 Name: <input type="text" ng-model="tweet.name"><br>
8 Tweet: <input type="text" ng-model="tweet.text"><br>
9 <br>
10 Preview: {{'"+ tweet.text + '" said '+ tweet.name}}
11
12 </div>
13
14 <script src="./angular.js"></script>
15
16 </body>
17 </html>
```



MyTweetApp

Name:

Tweet:

Preview: "Cool" said John

Being MEAN (client-side)

■ Connecting client and server

```
6 <div ng-app="" ng-controller="tweetController">
7   Name: <input type="text" ng-model="tweet.name"><br>
8   Tweet: <input type="text" ng-model="tweet.text"><br>
9   <br>
10  Preview: {{ "'" + tweet.text + '" said ' + tweet.name }}
11
12  <p>Your last tweets:</p>
13  <ul>
14    <li ng-repeat="tweet in tweets">
15      {{ tweet.text }}
16    </li>
17  </ul>
18
19 </div>
20
21 <script>
22 function tweetController($scope, $http) {
23   $scope.tweet = {};
24   $scope.tweet.name= "John";
25   $scope.tweet.text = "";
26
27   $Http.get("./tweets/"+$scope.tweet.name).success(function(response) {
28     $scope.tweets = response;
29   });
30 }
31
32 </script>
```



MyTweetApp

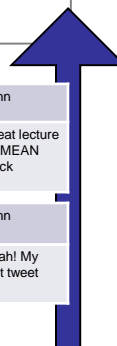
Name:

Tweet:

Preview: "Cool" said John

Your last tweets:

- Great lecture on MEAN stack
- Yeah! My first tweet



sender	John
text	Great lecture on MEAN stack
sender	John
text	Yeah! My first tweet

