BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Backend – Vertiefung II

Multimedia Engineering II
WS 13/14

Sven Spielvogel

Backend - Vertiefung

- Was ist eine API?
- Welche Merkmale besitzt eine API und welchen Einfluss haben diese?
- Welche negativen Aspekte gibt es in Bezug auf eine API zu beachten?
- Was bedeutet RESTful?
- Welche „Regeln“ gibt es für eine RESTful-Schnittstelle?
- Ist JSON immer kleiner als XML?
- Was ist JSON?
- Welche Vorteile hat JSON gegenüber XML?

Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Datenbankvergleich
  - MongoDB
  - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

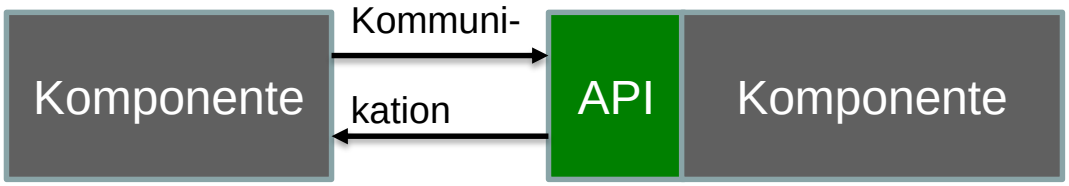
Letzte Vorlesung

Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Grundbegriffe
  - API
  - REST
  - JSON
- Programmierung – PHP
  - Kurzer Überblick
- Datenbankvergleich
    - MongoDB
    - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

API

- **A**pplication **P**rogramming **I**nterface
  - Schnittstelle zur Anbindung verschiedener Hard- und Software-Komponenten untereinander
  - Ermöglicht die Kommunikation und dient somit als Bindeglied



REST

- Representational State Transfer
  - Einheitliche Uniforme Schnittstelle
  - Lesbares Targeting in der URL
- Regeln
  - Plural
  - Spezifisch
  - Eindeutige URL
  - Suchparameter mit ?
  - Zustandslos
  - HTTP Codes
  - Versionierung
  - Pagination
  - Ausgabeformat XML/JSON/…

REST – Ein paar „Regeln“

- HTTP-Operationen (GET,PUT,POST,DELETE)

Ziel	POST create	GET read	PUT update	DELETE delete
`/orders`	Neue Bestellung erstellen	Auflisten der Bestellungen	Alle Bestellungen aktualisieren	Alle Bestellungen löschen
`/orders/1`	Fehler	Details für Bestellung 1	Aktualisieren - wenn nicht vorh. Fehler	Bestellung 1 löschen

- **J**ava**S**cript **O**bject **N**otation
  - Syntax zum Austausch und Speichern von Text-Informationen
  - Sprachunabhängig
  - Selbstbeschreibend
  - schneller und leichter zu parsen als XML
  - JSON kann größer sein als XML*
  - Parser in nahezu jeder Programmiersprache

* http://wiki.hl7.org/index.php?title=JSON_based_ITS

JavaScript Object Notation

```
{
    "employees": [
        { "firstName":"John" , "lastName":"Doe" },
        { "firstName":"Anna" , "lastName":"Smith" },
        { "firstName":"Peter" , "lastName":"Jones" }
    ]
}
```

- Zusammenfassung
  - schnell
    - http://jsperf.com/domparser-vs-json-parse/4
  - gut lesbar
  - schmal
  - ein String
  - kompatibel

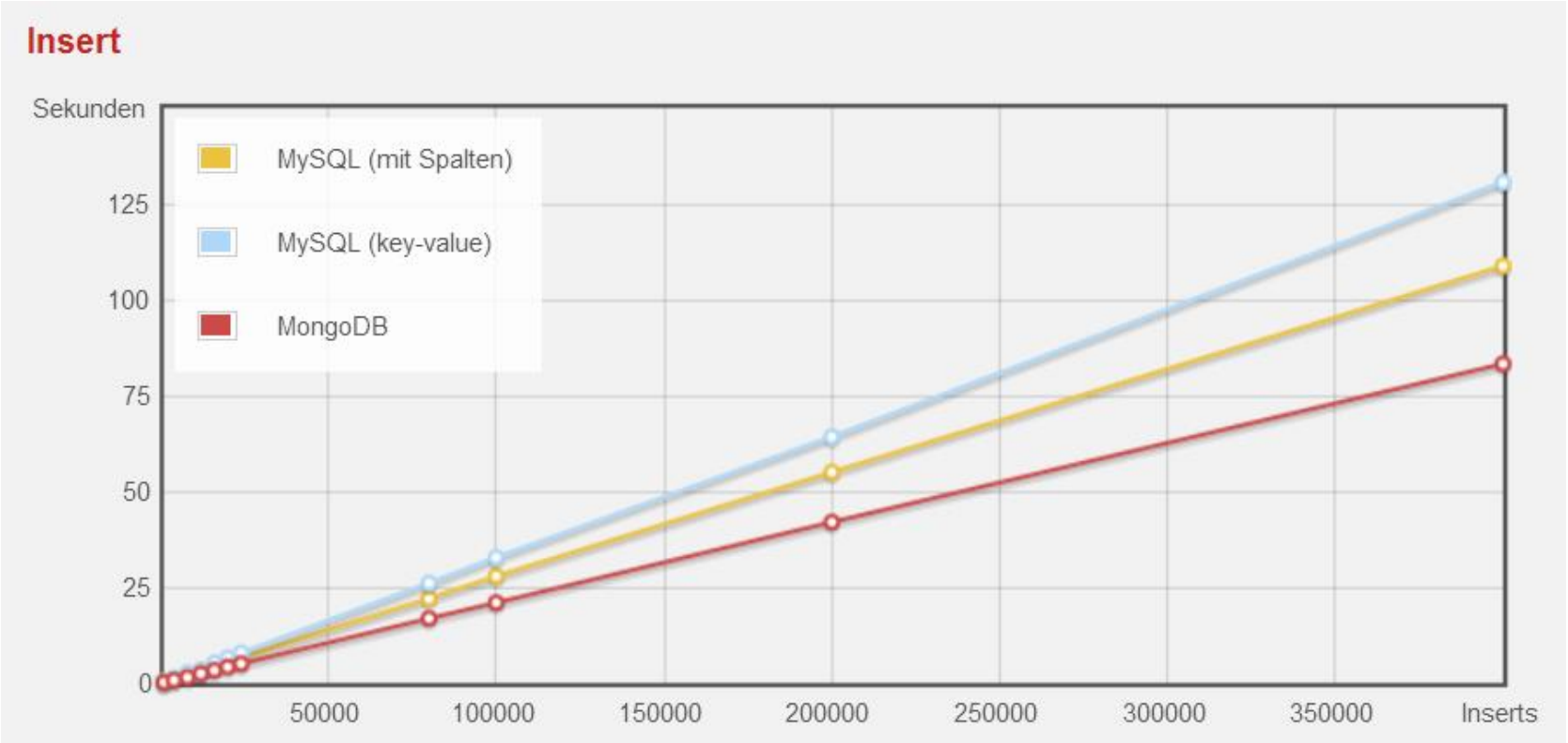
Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Datenbankvergleich
  - MongoDB
  - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

VS

- Voraussetzungen
  - Python
  - MongoDB
  - MySQL
    - 1. Jeder Key in eigene Spalte
    - 2. Key-Value(JSON)

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

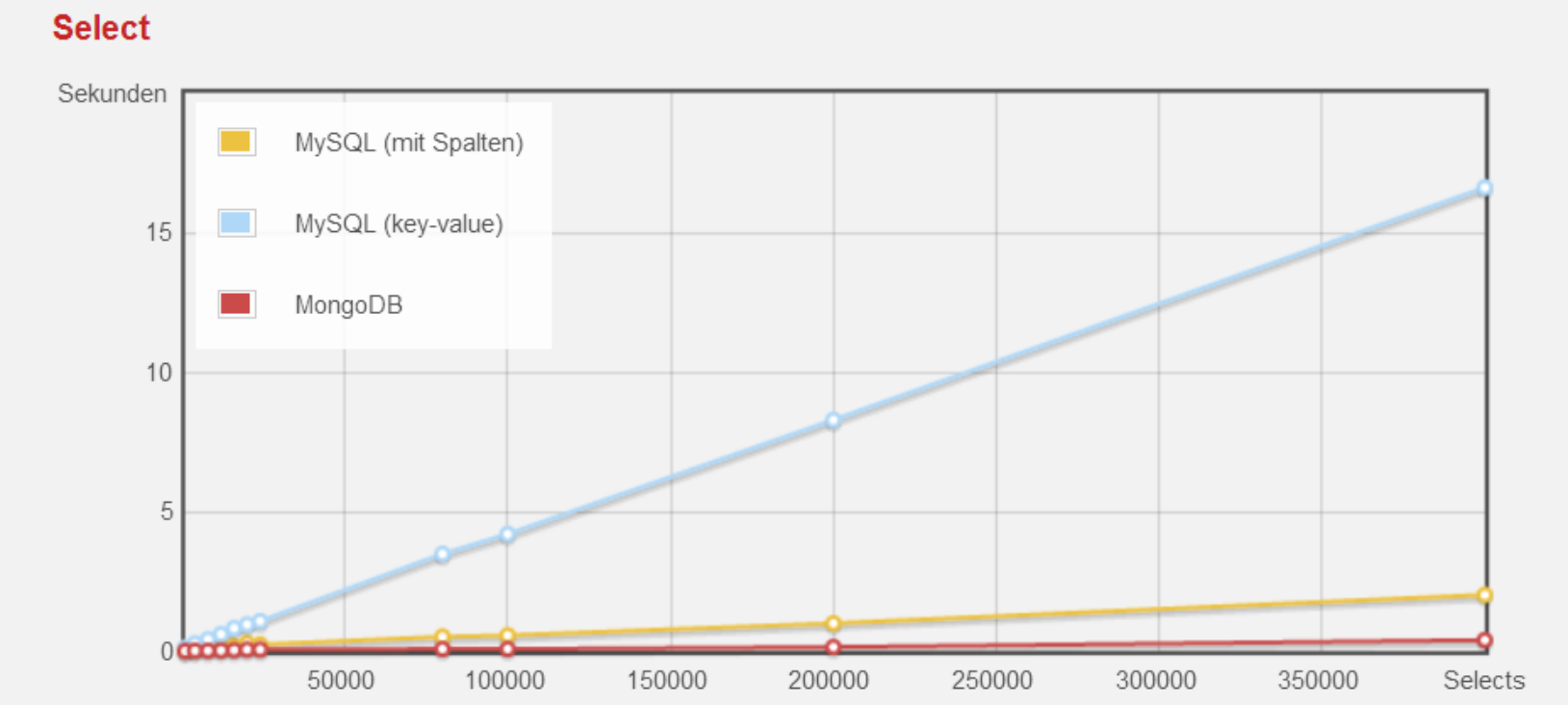


MongoDB 30,6% schneller als MySQL

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

Anzahl Elemente	MySQL (mit Spalten)	MySQL (Key-Value)	MongoDB
1000	0,28 s	0,31 s	0,16 s
4000	1,1 s	1,3 s	0,74 s
8000	2,2 s	2,67 s	1,43 s
12000	3,29 s	3,2 s	2,42 s
16000	4,4 s	5,23 s	3,3 s
20000	5,49 s	6,54 s	4,18 s
24000	6,62 s	7,84 s	5,0 s
80000	22,04 s	26,07 s	16,82 s
100000	27,85 s	32,74 s	20,91 s
200000	54,99 s	64,24 s	41,94 s
400000	108,8 s	130,69 s	83,26 s

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



MongoDB 500% schneller als MySQL

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

Anzahl Elemente	MySQL (mit Spalten)	MySQL (Key-Value)	MongoDB
1000	0,02 s	0,12 s	<0,01 s
4000	0,05 s	0,27 s	0,01 s
8000	0,12 s	0,42 s	0,01 s
12000	0,14 s	0,6 s	0,02 s
16000	0,17 s	0,82 s	0,03 s
20000	0,28 s	0,95 s	0,05 s
24000	0,23 s	1,07 s	0,05 s
80000	0,51 s	3,48 s	0,07 s
100000	0,56 s	4,19 s	0,07 s
200000	0,99 s	8,28 s	0,14 s
400000	2,01 s	16,61 s	0,39 s

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

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 _idfield.
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	MongoDB Schema Statements
CREATE TABLE users (id MEDIUMINT **NOT NULL** AUTO_INCREMENT, user_id Varchar(30), age Number, status char(1), **PRIMARY KEY** (id))	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")
ALTER TABLE users **ADD** join_date DATETIME	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/

ALTER TABLE users **DROP COLUMN** join_date	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** })
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	MongoDB insert() Statements
`INSERT INTO users(user_id, age, status) VALUES ("bcd001", 45, "A")`	`db.users.insert( { user_id: "bcd001", age: 45, status: "A" } )`

SQL Update Statements	MongoDB update() Statements
`UPDATE users SET status = "C" WHERE age > 25`	`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	MongoDB remove() Statements
DELETE FROM users **WHERE** status = "D"	**db**.**users**.**remove**({ **status**: "D" })
DELETE FROM users	**db**.**users**.**remove**( )

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

SQL SELECT Statements	MongoDB find() Statements
SELECT * **FROM** users	**db.users.find**()
SELECT id, user_id, status **FROM** users	**db.users.find**({ }, { **user_id**: 1, **status**: 1 })
SELECT user_id, status **FROM** users	**db.users.find**({ }, { **user_id**: 1, **status**: 1, **_id**: 0 })
SELECT * **FROM** users **WHERE** status = "A"	**db.users.find**({ **status**: "A" })
SELECT user_id, status **FROM** users **WHERE** status = "A"	**db.users.find**({ **status**: "A" }, { **user_id**: 1, **status**: 1, **_id**: 0 })
SELECT * **FROM** users **WHERE** status != "A"	**db.users.find**({ **status**: { **$ne**: "A" } })
SELECT * **FROM** users **WHERE** status = "A" **AND** age = 50	**db.users.find**({ **status**: "A", **age**: 50 })
SELECT * **FROM** users **WHERE** status = "A" **OR** age = 50	**db.users.find**({ **$or**: [{ **status**: "A" } , { **age**: 50 }] })

Viele weitere Beispiele siehe MongoDB-Referenz

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

Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Datenbankvergleich
  - MongoDB
  - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

Slim

PHP Micro Framework

Slim

```
<?php
$app = new \Slim\Slim();

$app->get('/hello/:name', function ($name) {
    echo "Hello, $name";
});

$app->run();
?>
```

Slim

- Powerful router
  - Standard and custom HTTP methods
  - Route parameters with wildcards and conditions
  - Route redirect, halt, and pass
  - Route middleware
- Template rendering with custom views
- Flash messages
- Secure cookies with AES-256 encryption
- HTTP caching
- Logging with custom log writers
- Error handling and debugging
- Middleware and hook architecture
- Simple configuration

http://docs.slimframework.com/

Slim

- Erzeugen einer Instanz

```
$app = new \Slim\Slim();
```

- Anlegen einer GET Route

```
$app->get('/hello/:name', function ($name) {
    echo "Hello, $name";
});
```

- Slim Applikation starten

```
$app->run();
```

Slim

```
// Library group
    $app->group('/library', function () use ($app) {
        // Get book with ID
        $app->get('/books/:id', function ($id) {

        });

        // Update book with ID
        $app->put('/books/:id', function ($id) {

        });

        // Delete book with ID
        $app->delete('/books/:id', function ($id) {

        });
    });
```

```
GET /path_to_api/library/books/1
```

```
DELETE /path_to_api/library/books/1
```

http://docs.slimframework.com/

Slim

- Routen über Gruppen bilden, wenn möglich
- Regeln anwenden!
- HTTP-Methoden

```
$app->get('/books/:id', function ($id) {
});
```

- Achtung! Rewrite Rule für die API!!

http://docs.slimframework.com/#Route-URL-Rewriting

```
GET /path_to_api/index.php/library/books/1
```

```
GET /path_to_api/library/books/1
```

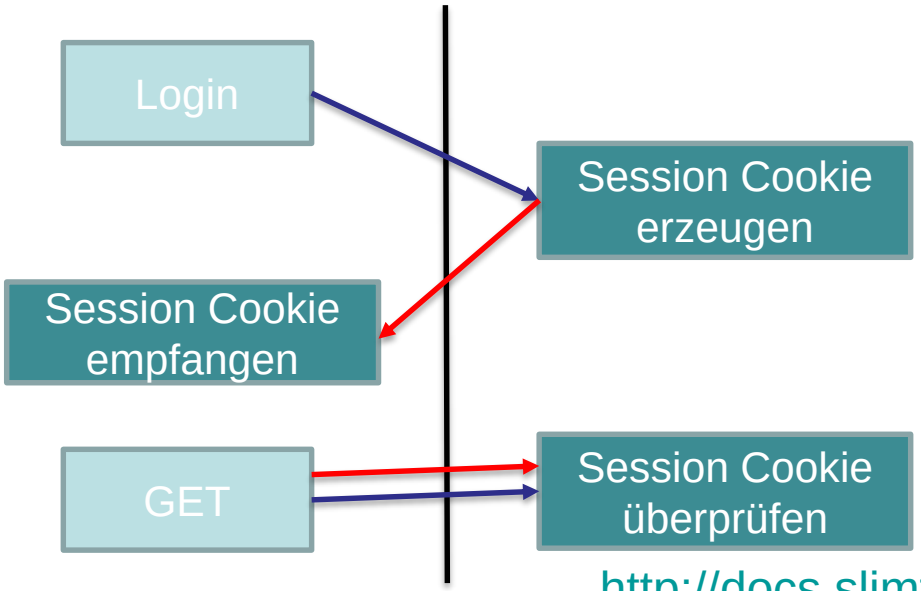
http://docs.slimframework.com/

Slim – Session Cookie

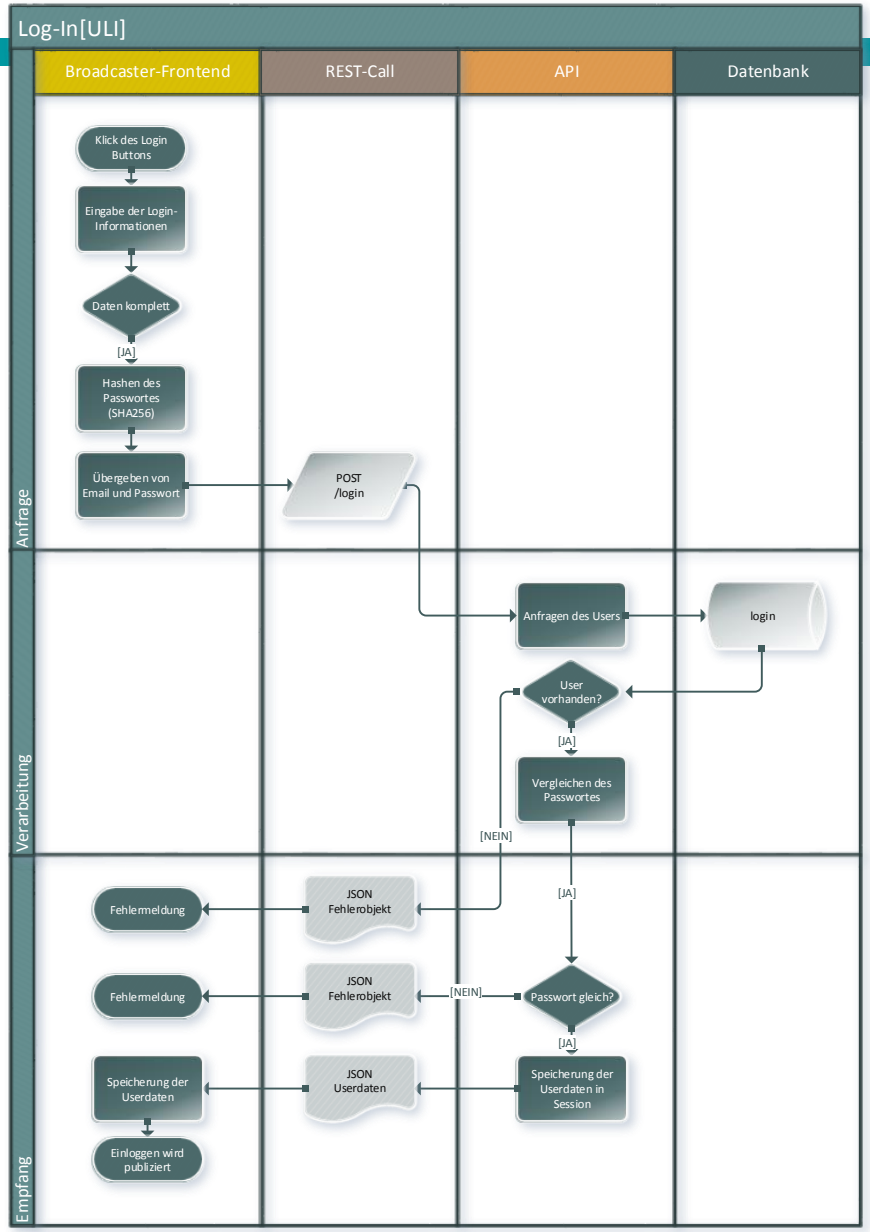
- Herausforderung:
  - Zustandslos!

  → Keine Serverseitige Speicherung von Zuständen

Authentifizierung mit dem Server



http://docs.slimframework.com/


Log-In[ULI]
Broadcaster-Frontend
REST-Call
API
Datenbank
Anfrage
Verarbeitung
Empfang
Klick des Login Buttons
Eingabe der Login-Informationen
Daten komplett
[JA]
Hashen des Passwortes (SHA256)
Übergeben von Email und Passwort
POST /login
Anfragen des Users
login
User vorhanden?
[JA]
Vergleichen des Passwortes
[NEIN]
[JA]
Passwort gleich?
[NEIN]
[JA]
JSON Fehlerobjekt
JSON Fehlerobjekt
JSON Userdaten
Fehlermeldung
Fehlermeldung
Speicherung der Userdaten
Einloggen wird publiziert
Speicherung der Userdaten in Session

Slim – Session Cookie

- Middleware

```
$app->add(new \Slim\Middleware\SessionCookie(array(
    'expires' => '60 minutes',
    'path' => '/',
    'domain' => null,
    'secure' => false,
    'httponly' => false,
    'name' => 'my_user',
    'secret' => 'any_secret',
    'cipher' => MCRYPT_RIJNDAEL_256,
    'cipher_mode' => MCRYPT_MODE_CBC
)));
```

http://docs.slimframework.com/#Middleware-Overview

http://docs.slimframework.com/#Cookie-Session-Store

Slim – Session Cookie

- Daten speichern

```
$_SESSION['my_user'] = $user;
```

- Daten lesen

```
$user = $_SESSION['my_user'];
```

ACHTUNG! Max ~4KByte

Slim - Logging

- Logging-Modul
  - Ermöglicht eigenen Logger

  → z.B. Speichern in DB, Datei, usw.
  - Fertige Logger werden angeboten

```
//configure logging
$app = new \Slim\Slim(array(
    'log.writer' => $dateLogger,
    'debug' => true,
    'log.enabled' => true,
    'log.level' => \Slim\Log::DEBUG
));
```

Slim - Logging

- \Slim\Log::EMERGENCY
  - Level 1
- \Slim\Log::ALERT
  - Level 2
- \Slim\Log::CRITICAL
  - Level 3
- \Slim\Log::ERROR
  - Level 4
- \Slim\Log::WARN
  - Level 5
- \Slim\Log::NOTICE
  - Level 6
- \Slim\Log::INFO
  - Level 7
- \Slim\Log::DEBUG
  - Level 8

Slim - Logging

- Logger holen

```
$log = \Slim\Slim::getInstance()->getLog();
```

- Logging mit Level

```
$log->info("loginUser::password incorrect");
```

Slim – Und jetzt?

- PHP-Server
- mcrypt(siehe Installation – Backend Einführung)
  - Nur, wenn verschlüsselte Cookies benutzt werden
- Slim installieren
  - http://slimframework.com/install
- Los geht‘s!
  - http://docs.slimframework.com/

express 3.0.0

web application framework for node

Express

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

app.get('/hello/:name', function(req, res){
  res.send('hello , + req.params.name);
});

app.listen(3000);
```

Web Applications
Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.

APIs
With a myriad of HTTP utility methods and Connect middleware at your disposal, creating a robust user-friendly API is quick and easy.

Performance
Express provides a thin layer of features fundamental to any web application, without obscuring features that you know and love in node.js

Slim

- Erzeugen einer Instanz

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

- Anlegen einer GET Route

```
app.get('/hello/:name', function(req, res){
  res.send('hello , + req.params.name);
});
```

- Express Server starten

```
app.listen(3000);
```

Express Session Cookie

- `cookieSession()`
  - `req.session`-Objekt
- Optionen
  - `key` cookie name defaulting to connect.sess
  - `secret` prevents cookie tampering
  - `cookie` session cookie settings, defaulting to { path: '/', httpOnly: true, maxAge: null }
  - `proxy` trust the reverse proxy when setting secure cookies (via "x-forwarded-proto")

```
app.use(express.cookieSession());
```

http://expressjs.com/api.html#cookieSession

Express – Session Cookie

- Beispiel
  - https://gist.github.com/visionmedia/1491756

```
var express = require('express')
  , cookieSessions = require('./cookie-sessions');

var app = express();

app.use(express.cookieParser('manny is cool'));
app.use(cookieSessions('sid'));

app.get('/', function(req, res){
  req.session.count = req.session.count || 0;
  var n = req.session.count++;
  res.send('viewed ' + n + ' times\n');
})

app.listen(3000);
```

```
module.exports = function(name) {
  return function(req, res, next) {
    req.session = req.signedCookies[name] || {};

    res.on('header', function(){
      res.signedCookie(name, req.session, { signed: true });
    });

    next();
  }
}
```

Express – Logging

- Da NodeJS auf JavaScript basiert, gilt das gleiche dafür, wie auch für JS-web-Applikationen
  - `console.log`
- Requests generell loggen über Middleware

```
// simple logger
app.use(function(req, res, next){
  console.log('%s %s', req.method, req.url);
  next();
});
```

Express – Logging

- Framework für Logging
  - Winston

```
var winston = require('winston');
```

```
var logger = new (winston.Logger)({
  transports: [
   new (winston.transports.Console)(),
   new (winston.transports.File)({ filename:
'somefile.log' })
  ]
});
```

```
logger.info('Hello again distributed logs');
```

https://github.com/flatiron/winston

Express – Und jetzt?

- NodeJS
- Express (siehe Installation – Backend Einführung)
  - `npm install express`
- Los geht‘s!
  - http://expressjs.com/

- http://www.heise.de/developer/artikel/REST-Webservices-mit-Node-js-Teil-2-Express-als-Anwendungsserver-1803626.html

REST API

Alternativen

Restler 3

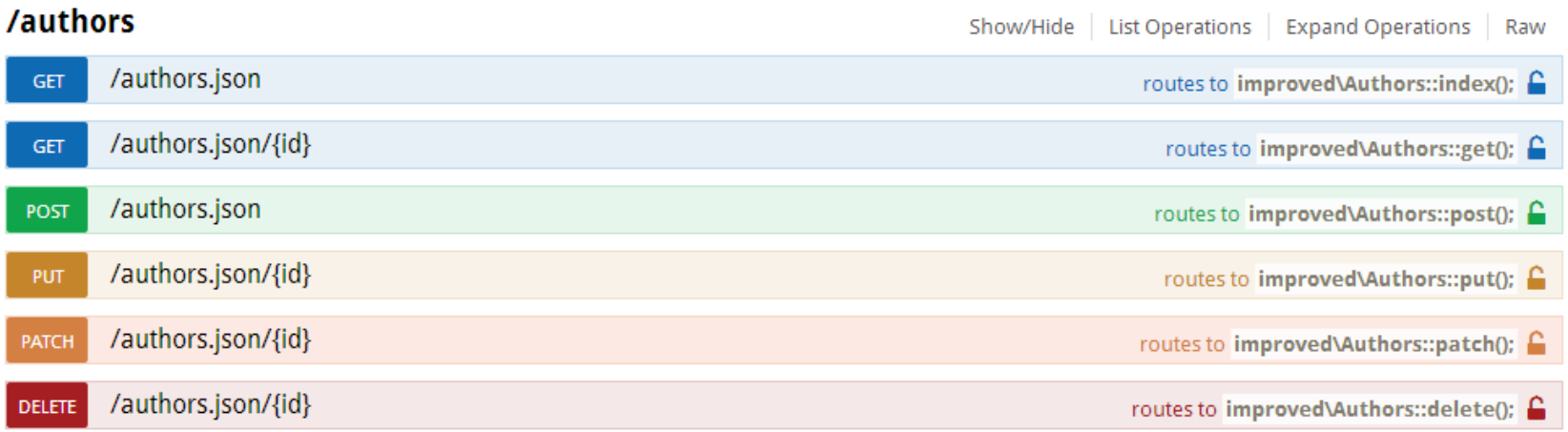
Restler 3

- PHP Rest Framework
  - Einfache OAuth2 Integration
  - Annotationen
    - PHPDoc

  → Automatische API Dokumentation
  - Gute Dokumentation des Frameworks
  - Viele Beispiele

Restler 3

- Automatische Dokumentation auf Basis der Annotationen



Restler 3

- Automatische Dokumentation auf Basis der Annotationen

```
/**
 * @param int $id
 *
 * @return array
 */
 function get($id)
   {
        $r = $this->dp->get($id);
        if ($r === false)
            throw new RestException(404);
        return $r;
   }
```

/authors

Show/Hide | List Operations | Expand Operations | Raw

GET /authors.json — routes to improved\Authors::index();

Implementation Notes

Add PHPDoc long description to **Authors::index();** (the api method) to write here

Parameters

Parameter	Value	Description

Try it out! Hide Response

Request URL

```
http://restler3.luracast.com/examples/_008_documentation/authors.json
```

Response Body

```
[
  {
    "id": 1,
    "name": "Jac Wright",
    "email": "jacwright@gmail.com"
  },
  {
    "id": 2,
    "name": "Arul Kumaran",
    "email": "arul@luracast.com"
  }
]
```

Response Code

```
200 OK
```

Response Headers

```
Pragma: no-cache
Date: Sat, 30 Nov 2013 11:52:43 GMT
Server: nginx/1.0.15
X-Powered-By: Luracast Restler v3.0.0rc4
Transfer-Encoding: chunked
Content-Language: en-US
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache, must-revalidate
Connection: keep-alive
Expires: 0
```

Restler 3

- http://restler3.luracast.com/

→ einfacher Start für eine professionelle API

- Umfangreicher und etwas schwieriger als mit SlimPHP

Testen einer API

Testen

- restify.io
- Postman
- curl
- Allgemeiner Ansatz
  - Testklassen

restify.io

- Online-Tool
- Schnelles und einfaches Testen
- Keine Auth-Verfahren (z.B. Oauth)


restify.io

Postman

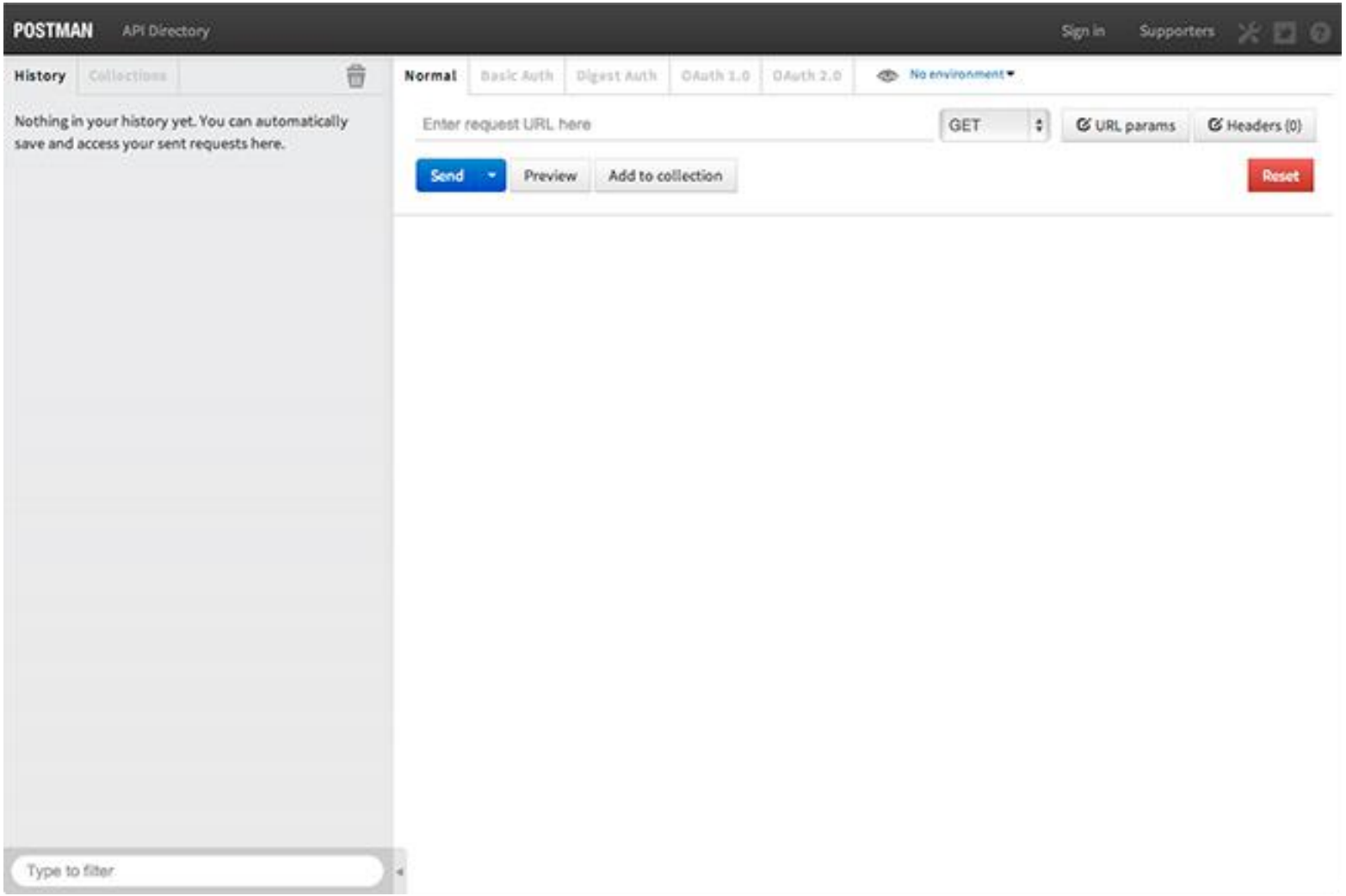
- Chrome Plug-In/App
- Sehr Umfangreich
- Viele Auth-Verfahren
  - Basic HTTP
  - Digest Auth
  - OAuth1.0
  - OAuth2.0
- History
- Proxy

http://www.getpostman.com/

Postman – Core Goals

- **Automate things as much as possible**
  - If there is something that can be solved using code, it should be solved using code
- **Never repeat yourself**
  - Testing APIs should not feel like endlessly entering data
- **The interface should adapt to the complexity of the request**
  - Only show what's relevant in the screen. Unroll the interface gradually
- **Complete control over your data**
  - Keep simple JSON based data formats which can be exported and shared as per user needs

http://www.getpostman.com/docs


POSTMAN
API Directory
Sign in
Supporters
History
Collections
Nothing in your history yet. You can automatically save and access your sent requests here.
Type to filter
Normal
Basic Auth
Digest Auth
OAuth 1.0
OAuth 2.0
No environment
Enter request URL here
GET
URL params
Headers (0)
Send
Preview
Add to collection
Reset

Postman – Core Goals

- **Automate things as much as possible**
  - If there is something that can be solved using code, it should be solved using code
- **Never repeat yourself**
  - Testing APIs should not feel like endlessly entering data
- **The interface should adapt to the complexity of the request**
  - Only show what's relevant in the screen. Unroll the interface gradually
- **Complete control over your data**
  - Keep simple JSON based data formats which can be exported and shared as per user needs

http://www.getpostman.com/docs

cURL http://curl.haxx.se/

- Konsolenbasiertes Tool
- Batch-Skripte möglich
- Ermöglicht alle möglichen Aufrufe
- Auth sehr schwierig

```
curl -i -H "Accept: application/json" -X PUT -d
"phone=1-800-999-9999"
http://192.168.0.165/persons/person/1
```

http://blogs.plexibus.com/2009/01/15/rest-esting-with-curl/

Allgemeiner Ansatz

- Testklassen
- Client-Seitig
- Komplexe Prozeduren
- Jedes Auth-Verfahren nach Implementierung

Ein Beispiel:

- http://pivotal.github.io/jasmine/

Allgemeiner Ansatz

- Bedeutung von Tests für eine API
  - Sicherstellung der Funktionalitäten
  - Sicherstellung der Qualität
  - Mögliche Referenz für Nutzer
  - Definition of Done

```
describe("A suite", function() {
        it("contains spec with an expectation", function() {
                expect(true).toBe(true);
        });
});
```

Jasmine

- Leichtgewichtig
- Sehr einfach in der Handhabung
- Intuitive Schreibweise der Tests
- Kann durch sämtliche Bibliotheken erweitert werden
  - jQuery
  - underscore.js
  - u.v.m.
- Einfache UI

```
Jasmine 1.3.0 revision 1354052693                finished in 9.157s

. . . . . .

Passing 10 specs                                 No try/catch

API Login/Logout
  should be able to request a session
  should be able to log out
  should be able to log in

API Channels
  it should be possible to fetch channels
  it should be possible to create a channel
  it should be possible to get a channel
  it should be possible to delete a channel

Encoder
  should be possible to set encoding url
  should be possible to start broadcasting
  should be possible to stop broadcasting
```

Jasmine

- Expectations
  - `toBe`
  - `toEqual`
  - `toMatch`
  - `toBeDefined`
  - `toBeUndefined`
  - `toBeNull`
  - `toBeTruthy`
  - `toBeFalsy`
  - …
  - `not` → `not.toBe` **oder** `not.toBeNull`

http://pivotal.github.io/jasmine/

Jasmine

- Setup

```
beforeEach(function() {
  foo = 0;
  foo += 1;
});
```

- Teardown

```
afterEach(function() {
  foo = 0;
});
```

http://pivotal.github.io/jasmine/

Jasmine API-Test

```
function getSession(callback) {
    $.ajax({
        type: "GET",
        url: "../../api/user/session",
        dataType: "json",
        success: callback
    });
}
```

Jasmine API-Test

- Spy

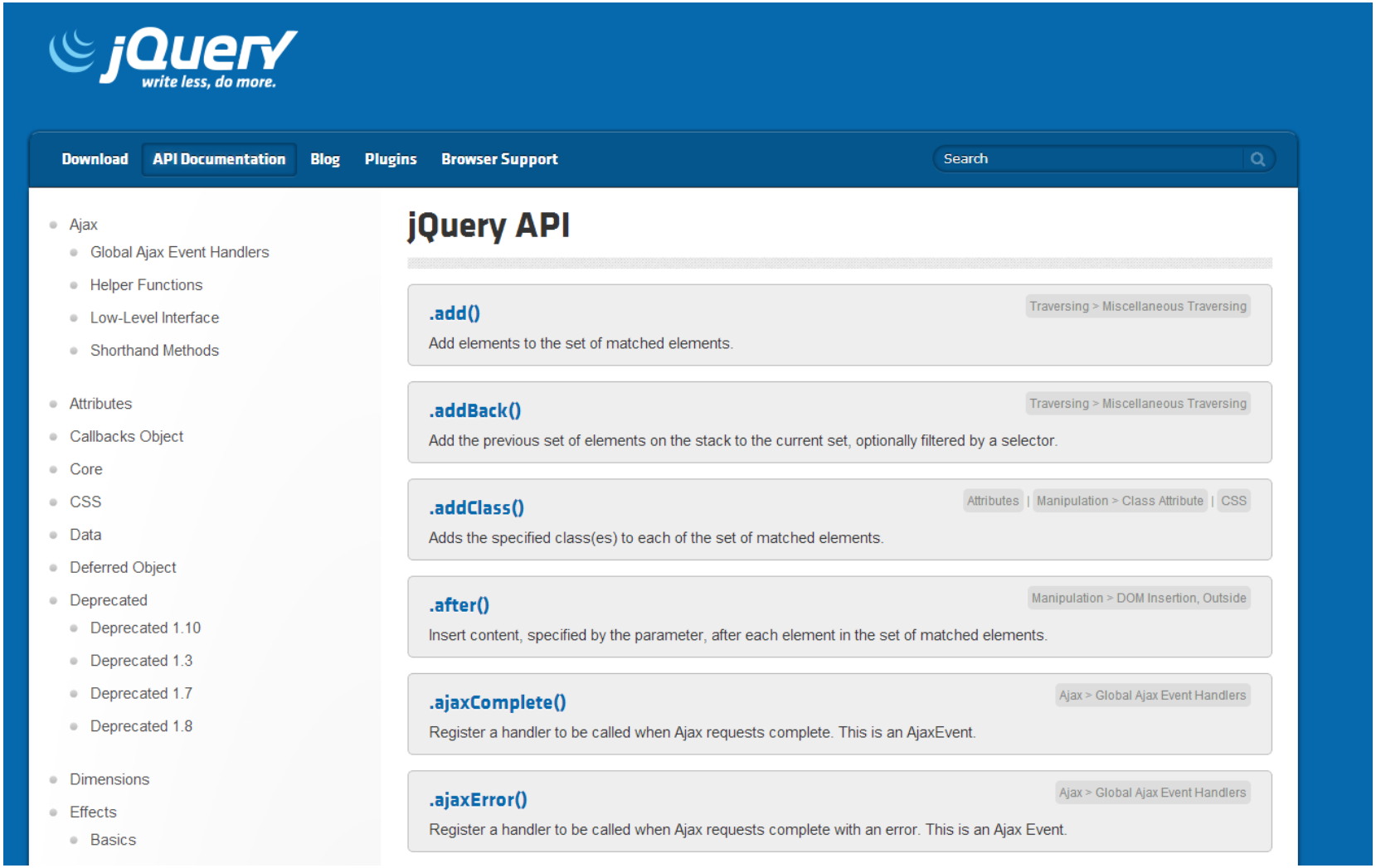
```
it("should be able to request a session", function () {
        var callback = jasmine.createSpy();
        getSession(callback);
        waitsFor(function() {
            return callback.callCount > 0;
        });
        runs(function() {
            expect(callback).toHaveBeenCalled();
        });
    });
```

Jasmine

- Weitere Funktionen → siehe Docs

  http://pivotal.github.io/jasmine/

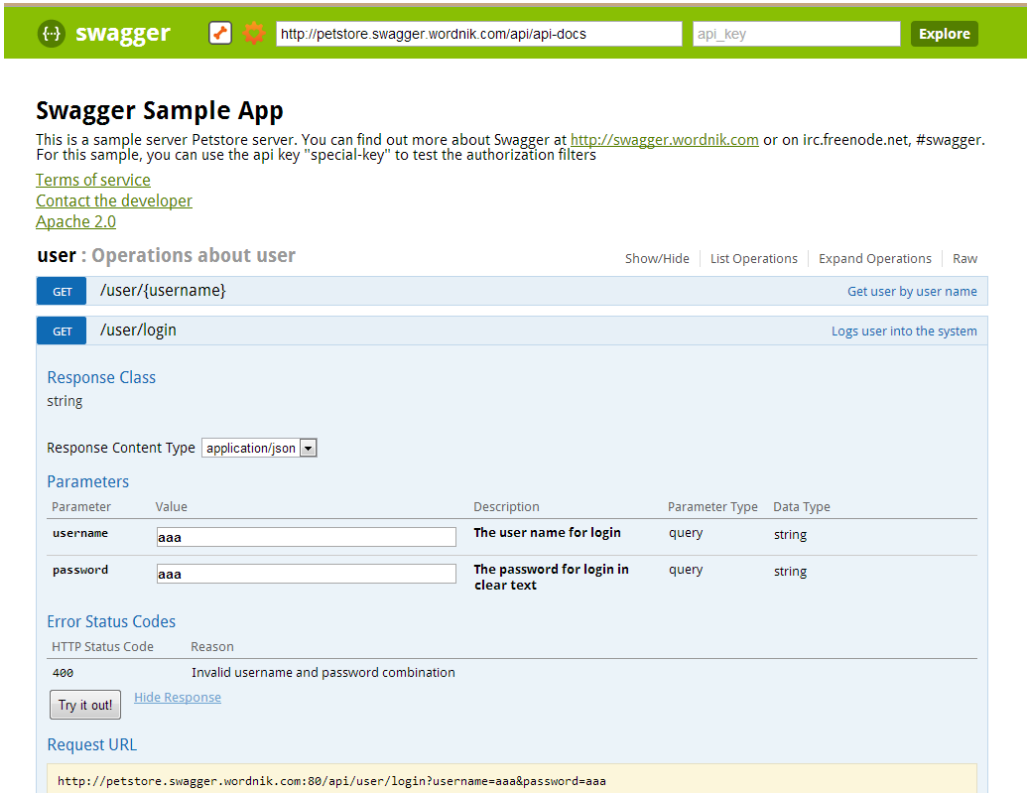
Dokumentation einer API


jQuery
write less, do more.
Download
API Documentation
Blog
Plugins
Browser Support
Search
Ajax
Global Ajax Event Handlers
Helper Functions
Low-Level Interface
Shorthand Methods
Attributes
Callbacks Object
Core
CSS
Data
Deferred Object
Deprecated
Deprecated 1.10
Deprecated 1.3
Deprecated 1.7
Deprecated 1.8
Dimensions
Effects
Basics
jQuery API
.add()
Traversing > Miscellaneous Traversing
Add elements to the set of matched elements.
.addBack()
Traversing > Miscellaneous Traversing
Add the previous set of elements on the stack to the current set, optionally filtered by a selector.
.addClass()
Attributes | Manipulation > Class Attribute | CSS
Adds the specified class(es) to each of the set of matched elements.
.after()
Manipulation > DOM Insertion, Outside
Insert content, specified by the parameter, after each element in the set of matched elements.
.ajaxComplete()
Ajax > Global Ajax Event Handlers
Register a handler to be called when Ajax requests complete. This is an AjaxEvent.
.ajaxError()
Ajax > Global Ajax Event Handlers
Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.

Einfluss der Dokumentation auf eine API

- Eine vernünftige Dokumentation macht eine API zu einer API
  - Direkter Zusammenhang zwischen Dokumentation und API
    - → Schlechte Doku – Schlechte API
- Eine gute Dokumentation muss nicht auf einem Framework basieren!
  - Word/PDF
  - Webseite
  - …

- Swagger
  - In Restler 3, kann aber auch so implementiert werden
  - https://github.com/wordnik/swagger-ui



- Webseite/PDF o.ä.

Type	Resource	Description
GET	/streams	returns all streams for authorized user
PUT	/stream/:id	saves specific stream
DELETE	/stream/:id	deletes specific stream
GET	/stream/:id	returns specific stream

GET /streams

returns all streams for authorized user

Dependency	- User logged in - Authorization level >= 0
Input	N/A
Output	JSON Array with stream objects
Example Input	N/A

Example Output	[
	{
	"id": "1",
	"account_id": "1bc9ece4-29ae-11e3-8094-736562617374",
	"name": "Webcam Übertragung",
	"type": "sv“
	}
	...

Vielen Dank für die Aufmerksamkeit!