Vorlesung

TECHNISCHE HOCHSCHULE LÜBECK

http:// RESTful API
GET PUT POST DELETE

Dart

Achtung: Mit Dart2 haben sich einige Änderungen ergeben. U.a. die Umstellung von optionaler auf statische Typisierung. Ggf. funktionieren daher einige Beispiele in den noch für Dart1 erstellten Folien nicht mehr. Wenn Sie so etwas finden, bitte sofort mailen!

Web-Technologien

Hypertext Transfer Protocol (HTTP)

Representational State Transfer (REST)

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
1

TECHNISCHE HOCHSCHULE LÜBECK

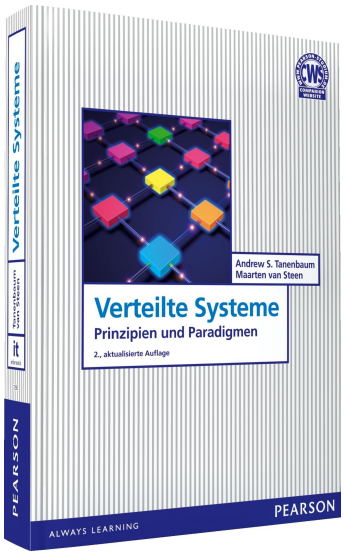
Prof. Dr. rer. nat.
Nane Kratzke
Praktische Informatik und
betriebliche Informationssysteme

- **Raum: 17-0.10**
- **Tel.:** 0451 300 5549
- **Email: nane.kratzke@th-luebeck.de**
- **Sprechzeiten: Mi. 14:00 bis 16:00**
  (nach Mail-Voranmeldung, oder jederzeit mit Termin)

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
2

Zum Nachlesen ...

TECHNISCHE HOCHSCHULE LÜBECK



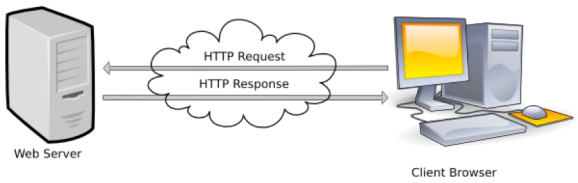
Chapter 5:
Representational State Transfer (REST)

5.1 Deriving REST
5.2 REST Architectural Styles
5.3 REST Architectural Views

HTTP in a Nutshell ...

TECHNISCHE HOCHSCHULE LÜBECK

- **Hypertext Transfer Protocol (HTTP)** ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht.
- Es wird hauptsächlich eingesetzt, um Webseiten (HTML-Dokumente) mit einem Webbrowser zu laden.
- Es ist jedoch nicht darauf beschränkt und kann auch als allgemeines Datenprotokoll eingesetzt werden (siehe bspw. REST).



https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol

URI – Uniform Resource Identifier

TECHNISCHE HOCHSCHULE LÜBECK

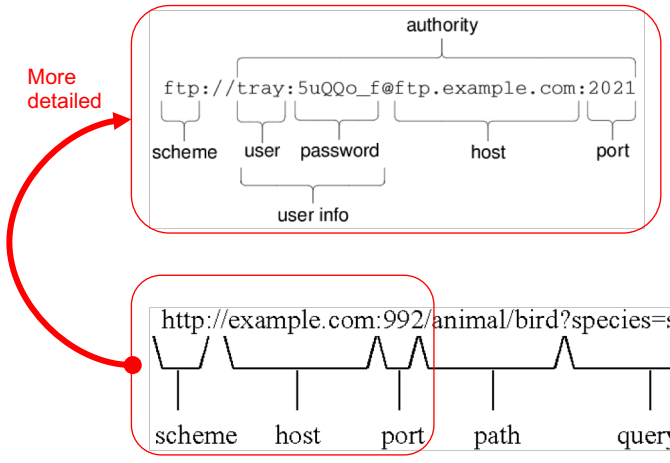
Ein Uniform Resource Identifier (Abk. URI) ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer Ressource (wie z.B. Webseiten, Dateien, Webservices, E-Mail-Empfängern, etc.) dient.

Name	Verwendung	Beispiel
http	HTTP	http://www.cs.vu.nl:80/globe
mailto	E-Mail	mailto:steen@cs.vu.nl
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Lokale Datei	file:/edu/book/work/chp/11/11
data	Eingefügte Daten	data:text/plain;charset=iso-8859-7,%e1%e2%e3
telnet	Anmeldung übers Netzwerk	telnet://flits.cs.vu.nl
tel	Telefon	tel:+31201234567
modem	Modem	modem:+31201234567;type=v32

Aus: Tanenbaum, van Steen; Verteilte Systeme; Pearson; Abb. 12.16 (Bsp. für URIs)

URL – Uniform Resource Locator

TECHNISCHE HOCHSCHULE LÜBECK

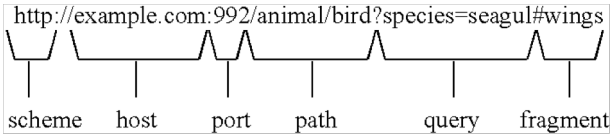


Ein Uniform Resource Locator (Abk. URL) ist eine spezielle Form einer URI, die vor allem im Webkontext gebräuchlich ist.

https://de.wikipedia.org/wiki/Uniform_Resource_Identifier

URL – Uniform Resource Locator

TECHNISCHE HOCHSCHULE LÜBECK



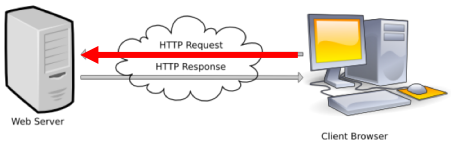
Uri
https://api.dartlang.org/stable/2.1.1/dart-core/Uri-class.html

```
Uri url = Uri.parse(
    "http://example.com:992/animal/bird?species=seagul#wings"
);

print("Scheme:    ${url.scheme}");
print("Authority: ${url.authority}");
print("Host:      ${url.host}");
print("Port:      ${url.port}");
print("Path:      ${url.path}");
print("Query:     ${url.query}");
print("Fragment:  ${url.fragment}");
```


HTTP Request

TECHNISCHE HOCHSCHULE LÜBECK



GET | POST | HEAD | PUT | DELETE | OPTIONS

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Request Line
Request Headers
Request Message Header
A blank line separates header & body
Request Message Body

HTTP Methoden

TECHNISCHE HOCHSCHULE LÜBECK

HTTP wurde als allgemein verwendbares Client-Server Protokoll entworfen. D.h. Ressourcen lassen sich nicht nur lesend (GET) von einem Server anfordern (was im Web sicherlich die am meisten verwendete Form ist, wenn man mit Webbrowsern „surft“), sondern auch erzeugen (POST), verändern (PUT), löschen (DELETE) bzw. nur Meta Informationen abrufen (HEAD).

Welche Methoden auf einer Ressource anwendbar sind, lässt sich mittels OPTIONS herausfinden.

Operation	Beschreibung
Head	Anforderung der Rückgabe eines Dokumentenkopfes
Get	Anforderung der Rückgabe eines Dokumentes zum Client
Put	Anforderung der Speicherung eines Dokumentes
Post	Verfügbarmachen von Daten, die einem Dokument (bzw. einer Sammlung) hinzugefügt werden sollen
Delete	Anforderung der Löschung eines Dokumentes

Aus: Tanenbaum, van Steen; Verteilte Systeme; Pearson; Abb. 12.11 (Von HTTP unterstützte Operationen/Methoden; nicht vollständig, umfasst nur die häufigsten Operationen)

HTTP Request
Erzeugen und Auswerten mit Dart

TECHNISCHE HOCHSCHULE LÜBECK

HTTPClientRequest
https://api.dartlang.org/stable/2.1.1/dart-io/HttpClientRequest-class.html

```dart
import 'dart:io';

[...]

final client = new HttpClient();
HttpClientRequest request =
  await client.get('www.th-luebeck.de', 80, '/index.html');

// Body schreiben
final body = "This is the message for the body";
request.contentLength = body.length;
request.write(body);

// Header
print("${request.method} ${request.uri}");
print(request.headers);

// Body
print("\n$body");
```

__Achtung:__ Wegen `dart:io` nur als Kommandozeilenapp und nicht im Browser lauffähig! Für Browser bitte `dart:html` und `HttpRequest` nutzen. Leicht andere API berücksichtigen.

HTTP Request (Ausgabe)

TECHNISCHE HOCHSCHULE LÜBECK

Vorherige Codezeilen würden folgende Konsolenausgabe erzeugen.

```
GET http://www.fh-luebeck.de/index.html
user-agent: Dart/1.13 (dart:io)
accept-encoding: gzip
content-length: 32
host: www.fh-luebeck.de

This is the message for the body
```

HTTP Request (Header Fields)

TECHNISCHE HOCHSCHULE LÜBECK

Aus: Tanenbaum, van Steen; Verteilte Systeme; Pearson; Abb. 12.13 (Einige HTTP-Nachrichtenköpfe)

Kopf	Quelle	Inhalt
Accept	Client	Die Dokumententypen, mit denen der Client umgehen kann
Accept-Charset	Client	Die für den Client akzeptablen Zeichensätze
Accept-Encoding	Client	Die Dokumentkodierungen, mit denen der Client umgehen kann
Accept-Language	Client	Die natürliche Sprache, mit denen der Client umgehen kann
Authorization	Client	Eine Liste der Anmeldeinformationen des Clients
WWW-Authenticate	Server	Sicherheitsaufforderung, die der Client beantworten muss
Date	Beide	Datum und Uhrzeit, zu der die Nachricht versandt wurde
ETag	Server	Die dem zurückgegebenen Dokument zugeordneten Tags
Expires	Server	Die Zeitdauer, für die die Antwort Gültigkeit behält
From	Client	Die E-Mail-Adresse des Clients
Host	Client	Der DNS-Name des Servers, auf dem sich das Dokument befindet
If-Match	Client	Die Tags, die das Dokument aufweisen sollte
If-None-Match	Client	Die Tags, die das Dokument nicht aufweisen sollte
If-Modified-Since	Client	Fordert den Server auf, ein Dokument nur dann zurückzugeben, wenn es nach dem angegebenen Zeitpunkt verändert wurde
If-Unmodified-Since	Client	Fordert den Server auf, ein Dokument nur dann zurückzugeben, wenn es nach dem angegebenen Zeitpunkt nicht mehr verändert wurde
Last-Modified	Server	Der Zeitpunkt der letzten Änderung des Dokumentes
Location	Server	Ein Dokumentenverweis, auf den der Client seine Anforderung umleiten sollte
Referer	Client	Bezieht sich auf das vom Client zuletzt angeforderte Dokument
Upgrade	Beide	Das Anwendungsprotokoll, zu dem der Absender wechseln will
Warning	Beide	Informationen zum Status der Daten in der Nachricht

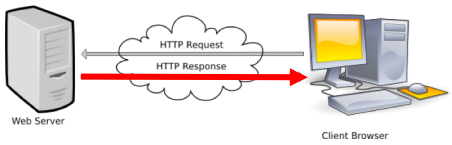
Über eine Reihe von Headern kann der Client dem Server mitteilen, welche Arten von Antworten er verarbeiten kann.

Umgekehrt, kann der Server dem Client über Header mitteilen, welche Art von Dokument ausgeliefert wurde.

https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

HTTP Response

TECHNISCHE HOCHSCHULE LÜBECK




```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

Status Line
Response Headers
Response Message Header
A blank line separates header & body
Response Message Body

Content Types

TECHNISCHE HOCHSCHULE LÜBECK

Typ	Untertyp	Beschreibung
Text	Plain	Unformatierter Text
	HTML	Text mit HTML-Auszeichnungsbefehlen
	XML	Text mit XML-Auszeichnungsbefehlen
Image	GIF	Standbild im GIF-Format
	JPEG	Standbild im JPEG-Format
Audio	Basic	Audio, 8-Bit PCM, Samplingrate 8000 Hz
	Tone	Ein bestimmter hörbarer Ton
Video	MPEG	Film im MPEG-Format
	Pointer	Darstellung eines Zeigegerätes für Präsentationen
Application	Octet-Stream	Eine uninterpretierte Bytefolge
	Postscript	Ein druckbares Dokument im PostScript-Format
	PDF	Ein druckbares Dokument im PDF-Format
Multipart	Mixed	Unabhängige Teile in der angegebenen Reihenfolge
	Parallel	Einzelbestandteile müssen gleichzeitig betrachtet werden

MIME = Multipurpose Internet Mail Exchange

Ursprünglich für Mail Protokolle gedacht, wird aber auch für Content-Types bei HTTP genutzt, um die Art des Dokumentes in der HTTP Response anzugeben.

Aus: Tanenbaum, van Steen; Verteilte Systeme; Pearson; Abb. 12.2 (Sechs MIME-Typen der allgemeinen Ebene und einige verbreitete Untertypen, es fehlt allerdings bspw. application/json)

HTTP Response
Erzeugen und Auswerten mit Dart

TECHNISCHE HOCHSCHULE LÜBECK

HTTPClientResponse
https://api.dartlang.org/stable/2.1.1/dart-io/HttpClientResponse-class.html

```
import 'dart:io';
import 'dart:convert';
[...]

final client = new HttpClient();
HttpClientRequest req =
  await client.get('www.th-luebeck.de', 80, '/index.html');

HttpClientResponse resp = await request.close();

// Response Header
print(
  "${req.method} ${req.uri} ${resp.statusCode} ${resp.reasonPhrase}"
);
print("${resp.headers}");
print("");

// Response Body
final body = await resp.transform(UTF8.decoder).join("\n");
print(body);
```

Achtung: Wegen `dart:io` nur als Kommandozeilenapp und nicht im Browser lauffähig! Für Browser bitte `dart:html` und `HttpRequest` nutzen. Leicht andere API berücksichtigen.

HTTP Response (Ausgabe)

TECHNISCHE HOCHSCHULE LÜBECK

Vorherige Codezeilen, würden folgende Konsolenausgabe erzeugen.

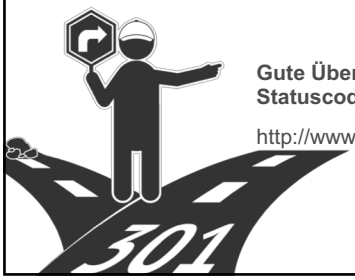
```
GET http://www.th-luebeck.de/index.html 200 OK
x-powered-by: PHP/5.4.45-0+deb7u2
cache-control: max-age=0
date: Mon, 25 Jan 2016 08:20:36 GMT
vary: Accept-Encoding
content-encoding: gzip
content-length: 8572
content-type: text/html; charset=utf-8
server: Apache
expires: Mon, 25 Jan 2016 08:20:36 GMT

<!DOCTYPE html>
<html lang="de">
<head>
[...]
```

HTTP Statuscodes

TECHNISCHE HOCHSCHULE LÜBECK

- **1xx** Informationen
- **2xx** Erfolgreiche Operation
- **3xx** Umleitung (Redirect)
- **4xx** Client Fehler (z.B. Bad Request, Not Found)
- **5xx** Server Fehler

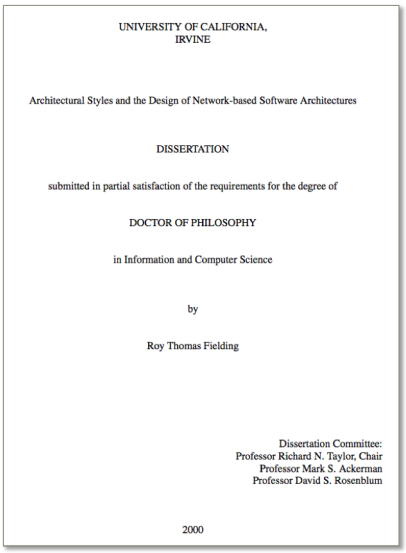


Gute Übersicht gebräuchlicher HTTP Statuscodes:

http://www.restapitutorial.com/httpstatuscodes.html

Zum Nachlesen ...

TECHNISCHE HOCHSCHULE LÜBECK



UNIVERSITY OF CALIFORNIA,
IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenblum

2000

Chapter 5:
Representational State Transfer (REST)

5.1 Deriving REST
5.2 REST Architectural Styles
5.3 REST Architectural Views

REST in a Nutshell ...

TECHNISCHE HOCHSCHULE LÜBECK

- **Representational State Transfer (REST)** ist ein Architekturstil für verteilte Systeme auf Basis von **HTTP**
- REST APIs dienen primär der **Maschine-Maschine-Kommunikation**
- REST ist eine Alternative zu ähnlichen Ansätzen wie SOAP oder RPC
- REST Requests arbeiten auf **Ressourcen** (anders als bspw. RPC)
- Client und Server können in anderen Sprachen realisiert sein **(language agnostic)**
- **REST Responses** nutzen daher gerne sprachunabhängige Serialisierungsformate wie bspw. JSON oder XML
- Die für REST nötige **Infrastruktur und Protokolle** sind im WWW bereits vorhanden
- Da das WWW (horizontal) skalierbar ist, sind es REST-basierte Systeme zumeist auch (wenn man ein paar Dinge beachtet)

https://de.wikipedia.org/wiki/Representational_State_Transfer

Für REST benötigte HTTP Methoden

TECHNISCHE HOCHSCHULE LÜBECK

Methode	Bedeutung	Idempotent	Sicher	CRUD
POST	Fügt eine neue Ressource hinzu			create
GET	Fordert eine Ressource an	x	x	read
PUT	Ändert eine Ressource	x		update
DELETE	Löscht eine Ressource	x		delete

Sicher (safe): Eine Operation ändert nicht den Zustand (nebeneffektsfreie Operation)

Idempotent: Wird eine Operation zweimal hintereinander ausgeführt, ist die zweite Operation sicher (d.h. ändert nichts mehr am Zustand)

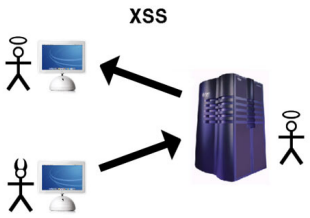
Same-Origin-Policy

TECHNISCHE HOCHSCHULE LÜBECK

Die Same-Origin-Policy (SOP) ist ein Sicherheitskonzept, das clientseitigen Skriptsprachen untersagt, auf Ressourcen zuzugreifen, deren Speicherort nicht dem Origin entspricht. Sie stellt ein Sicherheitselement in modernen Browsern zum Schutz vor Angriffen dar.

Hintergrund: Skriptsprachen im Browser haben Zugriff auf Kommunikation zwischen Browser und Web-Server. Dies beinhaltet sowohl das Auslesen, die Manipulation, das Empfangen und Senden von Daten.

Daraus ergibt sich die Sicherheitsanforderung, dass keine Daten aus einem Kontext (zum Beispiel der Verbindung des Browsers zu der Seite einer Bank) von einem Skript aus einem anderen Kontext zugreifbar oder manipulierbar sein darf. Um dies zu erreichen, wird beim Zugriff eines Skriptes auf eine Ressource die Herkunft (origin) von beiden verglichen. Der Zugriff wird nur erlaubt, wenn Skript und angeforderte Ressource dieselbe Herkunft (origin) haben (vom selben Server stammen).



WIKIPEDIA
Die freie Enzyklopädie

Quelle: https://de.wikipedia.org/wiki/Same-Origin-Policy

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
21

Same Origin Definition

TECHNISCHE HOCHSCHULE LÜBECK

http://www.example.com/dir/page.html

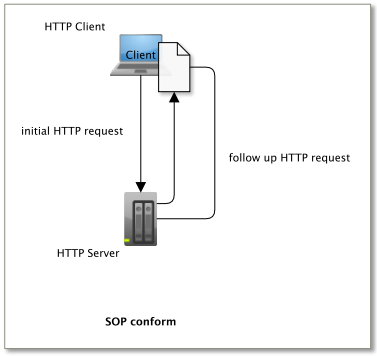
Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)

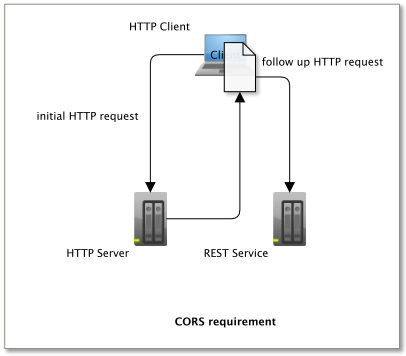
Die Same-Origin-Policy (SOP) ist erfüllt, wenn bei zwei Ressourcen deren URL-Anteile Scheme, Host und Port übereinstimmen.

Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
22

SOP und REST vertragen sich nicht immer

TECHNISCHE HOCHSCHULE LÜBECK





Die Same-Origin-Policy (SOP) ist dann ggf. hinderlich, wenn von einem Server eine Webapplikation geladen wird, die mit einem zweiten (nicht notwendig identischem) Server eine z.B. REST-basierte Kommunikation aufbauen muss.

In solchen Fällen kann der HTTP Server, der follow up requests bearbeitet, seine Responses mittels CORS Headers (Cross Origin Resource Sharing) kennzeichnen. Der HTTP Client verzichtet dann auf eine Same Origin Prüfung.

CORS
Ermöglichen mit Dart

TECHNISCHE HOCHSCHULE LÜBECK

```
/**
 * Sets CORS headers for HTTP responses.
 */
void enableCors(HttpResponse response) {
  response.headers.add(
      "Access-Control-Allow-Origin",
      "*"
  );
  response.headers.add(
      "Access-Control-Allow-Methods",
      "POST, GET, DELETE, PUT, OPTIONS"
  );
  response.headers.add(
      "Access-Control-Allow-Headers",
      "Origin, Content-Type, Accept, Charset"
  );
}
```

Cross-Origin Resource Sharing (CORS) ist ein Mechanismus, der Webclients Cross-Origin-Requests ermöglicht. Die Einschränkungen, die durch die SOP auferlegt sind, können vom Server aufgehoben werden. Der Server kann den Zugriff durch setzen entsprechender `Access-Control-Allow-*` Response Header erlauben/einschränken.

Ein einfacher REST-basierter Hello World Service

TECHNISCHE HOCHSCHULE LÜBECK

Methode	Resource	CRUD	Beschreibung	Beispiel
POST	/greet/:lang	Create	Erzeugt einen neuen Gruß	POST /greet/deutsch greet=hallo%20welt
GET	/greet/:lang	Read	Liest einen Gruß für Sprache :lang	GET /greet/deutsch
PUT	/greet/:lang	Update	Ändert einen Gruß für Sprache :lang	PUT /greet/deutsch Greet=Hallo%20Welt
DELETE	/greet/:lang	Delete	Löscht den Gruß der Sprache :lang	DELETE /greet/deutsch
GET	/greets	List	Listet alle im Service gespeicherten Grüße (Antwort als JSON Map)	GET /greets

Das REST-Prinzip soll exemplarisch an einem kleinen Hello World Service demonstriert werden, der mehrsprachige Gruß Ressourcen („Hello World“ Flosklen) verwaltet. Der Service soll um Grüße in fremden Sprachen ergänzt werden, einen Gruß in einer spezifischen Sprache ausgeben, Grüße bestehender Sprachen ändern und löschen können. Ferner soll er einen Überblick über alle hinterlegten Sprachen und zugeordneten Grüße liefern können.

Ein einfacher REST Server in Dart

TECHNISCHE HOCHSCHULE LÜBECK

```dart
// A simple REST based Hello World service.
class Server {

  // Service state. Default languages and greets (UTF-8 encoded!).
  var greets = {
    "dutch" : "Hello wereld",
    "english" : "Hello world",
    "french" : "Bonjour monde",
    "german" : "Hallo Welt",
    "italian" : "Ciao mondo",
    "portuguese" : "Olá mundo",
    "spanish" : "Hola mundo"
  };

  // Internally used for JSON pretty printing in JSON responses.
  final json = new JsonEncoder.withIndent('  ');

  // Resource Path definitions.
  final greetUrl  = new UrlPattern(r'/greet/(\w+)'); // /greet/:lang
  final greetsUrl = new UrlPattern(r'/greets');      // /greets

  [...]
}
```

Ein einfacher REST Server in Dart

TECHNISCHE HOCHSCHULE LÜBECK

```dart
import 'dart:io';
import 'package:route/server.dart';

[...]

/**
 * Starts the REST API server.
 */
Future<Router> serve({ip: '0.0.0.0', port: 4040}) async {
  final server = await HttpServer.bind(ip, port);
  final router = new Router(server)
    ..serve(greetUrl,  method: 'OPTIONS').listen(optionsGreet)
    ..serve(greetUrl,  method: 'POST'   ).listen(createGreet)
    ..serve(greetUrl,  method: 'GET'    ).listen(readGreet)
    ..serve(greetUrl,  method: 'PUT'    ).listen(updateGreet)
    ..serve(greetUrl,  method: 'DELETE' ).listen(deleteGreet)
    ..serve(greetsUrl, method: 'GET'    ).listen(listGreets);
  return new Future.value(router);
}
```


OPTIONS /greet/:lang

TECHNISCHE HOCHSCHULE LÜBECK

```dart
final router = new Router(server)
    ..serve(greetUrl,  method: 'OPTIONS').listen(optionsGreet)
    ..serve(greetUrl,  method: 'POST'   ).listen(createGreet)
    ..serve(greetUrl,  method: 'GET'    ).listen(readGreet)
    ..serve(greetUrl,  method: 'PUT'    ).listen(updateGreet)
    ..serve(greetUrl,  method: 'DELETE' ).listen(deleteGreet)
    ..serve(greetsUrl, method: 'GET'    ).listen(listGreets);
```

Es gibt diverse HTTP Client Libraries, die bei POST, DELETE und PUT vorher per OPTIONS anfragen, was für Methoden auf einer Ressource überhaupt zulässig sind.

Es macht für diese Fälle Sinn für jede Ressource einen entsprechenden OPTIONS Handler vorzusehen, ansonsten scheitert ein Request bereits beim BITTE BITTE sagen.


```dart
/**
 * Handles OPTIONS /greet/:lang requests
 * (used by the dart:html HttpRequest class).
 * This is necessary to handle HTTP client behaviors
 * (often an OPTIONS call is launched to determine
 * whether a POST, DELETE, PUT call is allowed).
 */
void optionsGreet(HttpRequest req) {
  HttpResponse res = req.response;
  enableCors(res);
  res.write("POST, GET, DELETE, PUT, OPTIONS");
  res.close();
}
```

POST /greet/:lang

TECHNISCHE HOCHSCHULE LÜBECK

```dart
/**
 * Handles POST /greet/:lang requests to create a new language specific greet.
 * Params: greet (greet in language :lang)
 */
void createGreet(HttpRequest req) {
  HttpResponse res = req.response;                      // To be created response
  enableCors(res);                                      // set CORS headers
  final lang = greetUrl.parse(req.uri.path).first;      // Language part of path
  req.transform(UTF8.decoder).join("\n").then((body) {  // req is stream (body)
    final params = Uri.parse("?$body").queryParameters; // body contains params

    if (greets.containsKey(lang)) {                     // lang already existing
      res.statusCode = HttpStatus.CONFLICT;             // 409
      res.reasonPhrase = "Already existing";
      res.close();                                      // Send response (error)
      return;
    }

    if (!params.containsKey('greet')) {                 // no greet parameter
      res.statusCode = HttpStatus.BAD_REQUEST;          // 400
      res.reasonPhrase = "Missing parameter";
      res.close();                                      // Send response (error)
      return;
    }

    greets[lang] = params['greet'];                     // Change server state

    res.statusCode = HttpStatus.CREATED;                // 201
    res.close();                                        // Send response (succes)
  });
}
```

greet=Hallo%20Welt

GET /greet/:lang

TECHNISCHE HOCHSCHULE LÜBECK

```dart
final router = new Router(server)
    ..serve(greetUrl,  method: 'OPTIONS').listen(optionsGreet)
    ..serve(greetUrl,  method: 'POST'   ).listen(createGreet)
    ..serve(greetUrl,  method: 'GET'    ).listen(readGreet)
    ..serve(greetUrl,  method: 'PUT'    ).listen(updateGreet)
    ..serve(greetUrl,  method: 'DELETE' ).listen(deleteGreet)
    ..serve(greetsUrl, method: 'GET'    ).listen(listGreets);
```

```dart
/**
 * Handles GET /greet/:lang requests to read
 * a language specific greet.
 */
void readGreet(HttpRequest req) {
  HttpResponse res = req.response;
  enableCors(res);
  final lang = greetUrl.parse(req.uri.path).first;
  if (!greets.containsKey(lang)) {
    res.statusCode = HttpStatus.NOT_FOUND;    // 404
    res.reasonPhrase = "Not Found";
    res.write("Not Found");
    res.close();            // Send response (error)
    return;
  }
  res.write(greets[lang]);  // Read Status
  res.close();              // Send response (success)
}
```

DELETE /greet/:lang

TECHNISCHE HOCHSCHULE LÜBECK

```
final router = new Router(server)
    ..serve(greetUrl,  method: 'OPTIONS').listen(optionsGreet)
    ..serve(greetUrl,  method: 'POST'   ).listen(createGreet)
    ..serve(greetUrl,  method: 'GET'    ).listen(readGreet)
    ..serve(greetUrl,  method: 'PUT'    ).listen(updateGreet)
    ..serve(greetUrl,  method: 'DELETE' ).listen(deleteGreet)
    ..serve(greetsUrl, method: 'GET'    ).listen(listGreets);
```

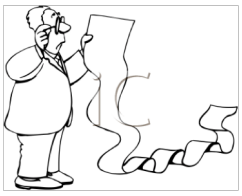
```
/**
 * Handles DELETE /greet/:lang requests to delete
 * a language specific greet.
 */
void deleteGreet(HttpRequest req) {
  HttpResponse res = req.response;
  enableCors(res);
  final lang = greetUrl.parse(req.uri.path).first;
  greets.remove(lang);            // Löschoperation
  res.close();
}
```

Eine DELETE Operation ist **idempotent**. D.h. wiederholte Löschung (bzw. das löschen nicht existenter Ressourcen) sollten nicht in einem Fehler enden.

GET /greets

TECHNISCHE HOCHSCHULE LÜBECK

```
final router = new Router(server)
  ..serve(greetUrl,  method: 'OPTIONS').listen(optionsGreet)
  ..serve(greetUrl,  method: 'POST'   ).listen(createGreet)
  ..serve(greetUrl,  method: 'GET'    ).listen(readGreet)
  ..serve(greetUrl,  method: 'PUT'    ).listen(updateGreet)
  ..serve(greetUrl,  method: 'DELETE' ).listen(deleteGreet)
  ..serve(greetsUrl, method: 'GET'    ).listen(listGreets);
```



```
/**
 * Handles GET /greets requests to list all
 * language specific greets (JSON).
 */
void listGreets(HttpRequest req) {
  HttpResponse res = req.response;
  enableCors(res);
  res.headers.contentType = new ContentType(
    "application", "json", charset: 'UTF-8'
  );
  res.write(json.convert(greets));
  res.close();
}
```

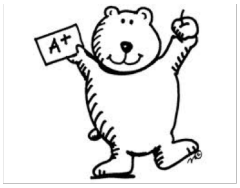
Liefert alle greets Ressourcen.

Zusammenfassung

TECHNISCHE HOCHSCHULE LÜBECK

- **HTTP**
  - Uniform Resource Identifiers (URI) und Locators (URL)
  - HTTP Request (Client) und Response (Server)
  - HTTP Methoden
  - HTTP Header
  - HTTP Content Types (MIME)
  - HTTP Status Codes
  - Dart Libraries für HTTP

- **Same Origin Policy (SOP) und Cross Origin Resource Sharing (CORS)**
- **REST**
  - CRUD und HTTP Methoden
  - Sichere und idempotente Operationen
  - REST am Bsp. eines Hello World Service
  - Dart Libraries für REST



Prof. Dr. rer. nat. Nane Kratzke
Praktische Informatik und betriebliche Informationssysteme
33