

Distributed Systems

RESTful APIs

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
<https://teaching.dahahm.de>

15.06.2023

Mistakes

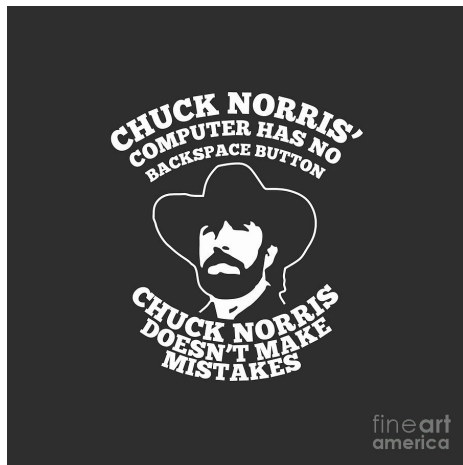
Who makes no mistakes?

- You
- Your professors
- The Pope
- Chuck Norris

Mistakes

Who makes no mistakes?

- You
- Your professors
- The Pope
- Chuck Norris



Evaluation

Please take some time to evaluate this course.
Your feedback is key to improve to improving teaching!



<https://umfragen.evasys.frankfurt-university.de/evasys/online.php?pswd=33V5T>

Exam, Preconditions, and Registration

- The **exam** will take place on **July 28, 2023**
 - The room and exact time will be announced in time
- Make sure you are registered for the *Prüfungsvorleistung*
- As soon as you have reached a sum of at least 50 points over all exercises, you have passed and are allowed to take part in the exam

What is a Web Service?

What is a Web Service?

Definition 1

“Web service is a software system designed to support interoperable machine-to-machine interaction over a network.”

– W3C, Web Services Glossary

What is a Web Service?

Definition 1

“Web service is a software system designed to support interoperable machine-to-machine interaction over a network.”

– W3C, Web Services Glossary

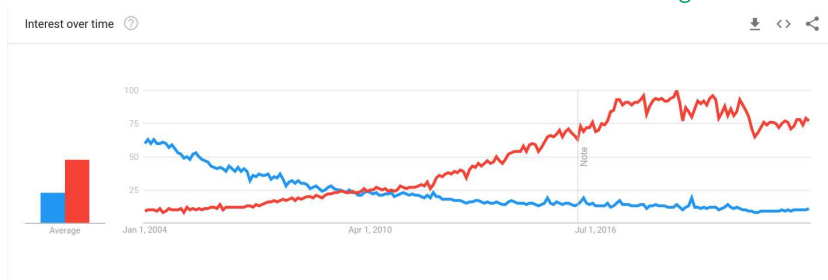
Definition 2

“We can identify two major classes of Web services: REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of “stateless” operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations.”

— W3C, Web Services Architecture (2004)

A Trend from SOAP to REST

Interest over time for REST API versus SOAP API based on Google Trends.



Agenda

- REST

- HTTP

- RESTful Web Services

Agenda

■ REST

■ HTTP

■ RESTful Web Services

Representational State Transfer

Representational State Transfer (REST) is an architectural style for distributed hypermedia systems.

Definition

"The name "Representational State Transfer" is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual **state-machine**), where the user progresses through the application by selecting links (**state transitions**), resulting in the next page (representing the **next state** of the application) being transferred to the user and rendered for their use."

Defined in 2000 by Roy Fielding in his doctoral dissertation.

State, Representation, Transfer

■ State

- All **stored data** of an application at a given **point of time**

State, Representation, Transfer

■ State

- All **stored data** of an application at a given **point of time**

■ Representation

- A representation contains all necessary information to modify a **resource**
- May be different from the server's internal representation
- The same data can have multiple representations (e.g., **JSON**, **HTML**, **XML**)
- All changes to the state of a resource shall only happen via its representation

State, Representation, Transfer

■ State

- All **stored data** of an application at a given **point of time**

■ Representation

- A representation contains all necessary information to modify a **resource**
- May be different from the server's internal representation
- The same data can have multiple representations (e.g., **JSON**, **HTML**, **XML**)
- All changes to the state of a resource shall only happen via its representation

■ Transfer

- Client-server architecture
- Server acts as a **producer**, stores and provides the data
- Client acts as a **consumer** and requests the data

What is REST?

- REST is an **architectural style**, not a standard
- It was designed for distributed systems to address **architectural properties** such as **performance, scalability, simplicity, modifiability, visibility, portability, and reliability**
- REST's architectural style is defined by **6 principles/architectural constraints** (→ next slide)
- Systems and APIs that conforms to the constraints of REST can be called **RESTful**

REST Principles

- Client-server

REST Principles

- Client-server
- Uniform interface
 - **Resource**-based
 - Manipulation of resource through representation
 - Self-descriptive messages
 - Hypermedia as the engine of application state

REST Principles

- Client-server
- Uniform interface
 - **Resource**-based
 - Manipulation of resource through representation
 - Self-descriptive messages
 - Hypermedia as the engine of application state
- **Stateless** interactions

REST Principles

- Client-server
- Uniform interface
 - **Resource**-based
 - Manipulation of resource through representation
 - Self-descriptive messages
 - Hypermedia as the engine of application state
- Stateless interactions
- Cacheable

REST Principles

- Client-server
- Uniform interface
 - **Resource**-based
 - Manipulation of resource through representation
 - Self-descriptive messages
 - Hypermedia as the engine of application state
- Stateless interactions
- Cacheable
- Layered system

REST Principles

- Client-server
- Uniform interface
 - **Resource**-based
 - Manipulation of resource through representation
 - Self-descriptive messages
 - Hypermedia as the engine of application state
- Stateless interactions
- Cacheable
- Layered system
- Code on Demand (optional)

Building RESTful API

- Can be build on top of existing **web technologies**
- Reusing semantics of **HTTP 1.1** methods
- **Safe** and **idempotent methods**
- Typically called HTTP verbs in context of services
- **Resource oriented**, correspond to **CRUD** operations
- Satisfies uniform interface constraint
- HTTP Headers to describe requests & responses
- **Content negotiation**

Agenda

■ REST

■ HTTP

■ RESTful Web Services

Hypertext Transfer Protocol (HTTP)

- HTTP is a **stateless** protocol for data transmission
 - Stateless means that every HTTP message contains all the information necessary to understand the message
 - The server does not maintain any information regarding the state or session for the client, and each request is a transaction, independent of other requests
- Specified in **RFC 1945**, **2068**, **7540**, and many more
- Uses **TCP** via **port 80** or **443** (HTTPS → HTTP over a secure channel)
- Originally developed by Roy Fielding, **Tim Berners-Lee**, and others at **CERN** from 1989 onwards
- Currently in version 2 (**HTTP/2**) since 2015
- The proposed successor **HTTP/3** is based on **QUIC**

World Wide Web

- Together with the concepts of [URL](#)¹ and [HTML](#)² it is the basis of the [World Wide Web \(WWW\)](#)
- Original main purpose: Loading web pages from [webserver](#) in a [browser](#)
- HTTP needs a reliable transport protocol → TCP
- Each HTTP message consists of:
 - [HTTP header](#): Includes among others information about the encoding, desired language, browser, and content type
 - [Body](#): Contains the payload, e.g., the HTML source code of a web page
- Today many application work on top of HTTP, e.g., using [web sockets](#)

¹URL = Uniform Resource Locator

²HyperText Markup Language

HTTP Methods

- The HTTP protocol provides several requests messages

Request	Description
PUT	Upload a new resource to the web server
GET	Request a resource from the web server
POST	Upload data to the web server in order to generate resources
DELETE	Erase a resource on the web server

- Other commands like HEAD or TRACE exist.

About state

HTTP is a stateless protocol. But via cookies in the header information, applications can be implemented which require state or session information because they assign user information or shopping carts to clients.

What is the Difference between PUT and POST?

PUT

POST

What is the Difference between PUT and POST?

PUT

- Requests for the **attached entity** (in the request body)

POST

- Request that the origin server **accept the attached entity**

What is the Difference between PUT and POST?

PUT

- Requests for the **attached entity** (in the request body)
- **Idempotent**

POST

- Request that the origin server **accept the attached entity**
- **Not idempotent**

What is the Difference between PUT and POST?

PUT

- Requests for the **attached entity** (in the request body)
- **Idempotent**
- Modify a **singular resource** that is a part of resources collection.

POST

- Request that the origin server **accept the attached entity**
- **Not idempotent**
- Add a **child resource** under **resources collection**

What is the Difference between PUT and POST?

PUT

- Requests for the **attached entity** (in the request body)
- **Idempotent**
- Modify a **singular resource** that is a part of resources collection.
- Use as **UPDATE**

POST

- Request that the origin server **accept the attached entity**
- **Not idempotent**
- Add a **child resource** under **resources collection**
- Use as **CREATE**

Safety and Idempotency

Which HTTP methods are idempotent?

Safety and Idempotency

HTTP Method	Safe	Idempotent
GET	✓	✓
PUT	✗	✓
DELETE	✗	✓
POST	✗	✗
HEAD	✓	✓
OPTIONS	✓	✓
TRACE	✓	✓
PATCH	✗	✗

HTTP Responses

- Each HTTP response contains a **status code**, which consists of three digits, and a text string, which describes the reason for the response

Status code	Meaning	Description
1xx	Informational	Request received, continuing process
2xx	Success operation	Action received, understood, accepted, and processed successfully
3xx	Redirection	Additional action must be taken by the client to complete the request
4xx	Client error	Request of the client caused an error situation
5xx	Server error	Server failed to fulfill a valid request ⇒ error was caused by server

Common HTTP Status Codes



Source: <http.cat>. Author: Tomomi Imura

- The table contains some common status codes of HTTP

Status code	Meaning	Description
200	OK	Request processed successfully. Result is transmitted in the response
204	No Content	Request executed successfully. Response intentionally contains no data
301	Moved Permanently	The old address is no longer valid
400	Bad Request	Request cannot be fulfilled due to bad syntax
401	Unauthorized	Request can not be executed without a valid authentication
403	Forbidden	Request is executed because of clients lack of privileges
404	Not Found	Server could not find the requested resource
500	Internal Server Error	Unexpected server error

HTTP Requests

- If an URL is accessed via HTTP (e.g., `http://example.teaching.dahahm.de/index.html`, the request for the resource `/index.html` is transmitted to the computer with hostname `example.teaching.dahahm.de`
- First, via DNS, the hostname is resolved to an IP address
- Next, this HTTP GET request is transmitted via TCP to port 80, where the web server usually operates

```
GET /index.html HTTP/1.1
Host: example.teaching.dahahm.de
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
...
```

Virtual Hosts (vhosts)

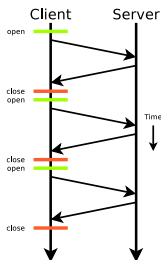
One server handles typically more than one domain, i.e., the same web server application may deliver multiple web pages at the same IP address for different domain names.

HTTP Response

- The HTTP response of the web server consists of a message header and the message body with the actual message
 - In this case, the message body contains the content of the requested file `index.html`

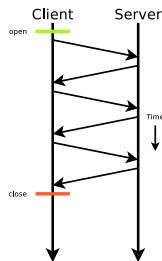
```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Fri, 28 Jan 2022 18:05:47 GMT
Content-Type: text/html
Content-Length: 274
Last-Modified: Fri, 28 Jan 2022 17:55:45 GMT
Connection: keep-alive
ETag: "61f42e21-112"
Accept-Ranges: bytes
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Example Page for teaching computer networks</title>
</head>
<body>
<p>Happy networking!</p>
</body>
</html>
```

HTTP Protocol Versions (HTTP/1.0 and HTTP/1.1)



- **HTTP/1.0** (RFC 1945): Prior to any request, a new TCP connection is established and closed by default by the server after the transmission of the reply

- **HTTP/1.1** (RFC 2616): By default, no connection termination is done
 - So the connection can be reused for multiple requests
 - Interrupted transmissions can be resumed with HTTP/1.1



HTTP Protocol Versions (HTTP/2)

- **HTTP/2** (RFC 7540): Changes from a text-based protocol to a **binary** one
 - Accelerates the data transfer by **compressing** the header with the **HPACK** algorithm (RFC 7541)
 - Enables the aggregation (*Multiplex*) of requests and a server can send (*Server Push*) data automatically, which it expects the browser to request immediately
 - Examples of such data are CSS files (Cascading Style Sheets), which specify the layout of web pages, or script files
 - Currently used by approx. 45% of all web servers
- **HTTP/3**: Is not yet an RFC
 - Based on **QUIC**
 - Currently used by approx. 20% of all web servers, but not yet supported by all browsers

Agenda

■ REST

■ HTTP

■ RESTful Web Services

Addressability

- Every “thing” has a URI
 - `http://sales.com/customers/323421`
 - `http://sales.com/customers/323421/address`
- A URI provides information about ...
 - The protocol (**how** do we communicate)
 - The host/port (**where** it is on the network)
 - The resource path (**what** resource are we communicating with)

RESTful Services

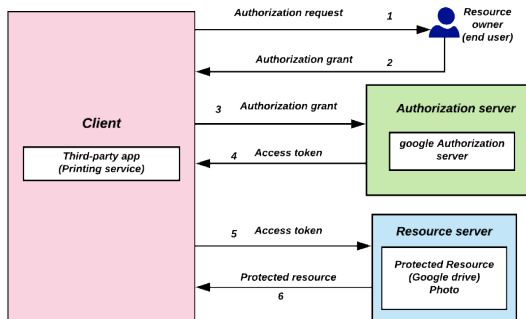
- Resources as URI
 - Use unique URI to reference every resource on your API
- Operations as HTTP Methods
 - GET – Queries
 - POST – Queries
 - PUT, DELETE – Insert, update, and delete
- Connectedness and Discoverability
 - Like the Web, HTTP Responses contains links to other resources

Authentication via OAuth

Open Authorization

OAuth is a standard specified by the IETF in RFC 5849 (OAuth 1.0) and RFCs 6749 and 6750 (OAuth 2.0) for access delegation and is often used to authorize access to REST APIs.

Abstract Flow



Important takeaway messages of this chapter

- Many modern distributed systems are built as web services using HTTP as a transport protocol.
- REST is an architectural style to design such a system
- In RESTful systems resources are represented as URIs and operations as HTTP methods

