FH Salzburg

VO Web-Technologien

Einheit 1, Oliver Jung

Technik
Gesundheit
Medien

Ihr LVA-Leiter

- Studium „Angewandte Informatik“ mit Schwerpunkt „Human-Computer Interaction“ an der PLUS
- Freiberuflicher Web-Entwickler
- Seit 2013 wissenschaftlicher Mitarbeiter (Forschung und Entwicklung) bei der Salzburg Research
- Projektleitung und Frontend-Entwicklung in vielzähligen nationalen und europäischen IKT-Forschungsprojekten
- https://www.salzburgresearch.at/

Organisatorisches

- Einteilung (2x 2 ECTS / 1,5 SWS)
  - Jeweils VO+LB (2 Gruppen) nacheinander
  - 9 Termine geblockt, im Schnitt alle 2 Wochen
  - VO: Prüfung am 08.01, Fragerunde/Vorbereitung am 11.12
  - LB: Aufeinander aufbauende Beispiele, 3-4 Abgaben, Ergebnispräsentation am 18.12

- Beurteilung
  - Abschlussklausur >= 50% und Labor >=50%
  - Anwesenheitspflicht >= 75%
  - Notenschlüssel (WIN, siehe Syllabus)

Inhalte VO

- Einführung, URIs und HTTP(S)
- HTML, CSS und Responsive Web Design
- Grundlagen und Konzepte der Web-Programmierung
  - Clientseitige Programmierung (insbesondere JavaScript)
  - Serverseitige Programmierung
- Exkurs: Frameworks und das Web von Morgen

Inhalte LB

- Entwurfsstrategien für Web-Applikationen
- HTML und CSS Basics
- Responsive Web Design
- Clientseitige Programmierung
- Authentifizierung
- Exkurs: Serverseitige Programmierung, Datenbanken, Logging und Testing
- ~~Frameworks~~

Hilfestellung

- **Heutige (und künftige) Slides**:
  - https://github.com/oliver-jung/lva-web-technologien-2024/tree/main

WWW – Ein Überblick

- **World Wide Web** ist eine weltweit verteilte Informations- und Datensammlung, auf die über das Internet mit Hilfe von **HTTP(S)** – Hyper Transfer Protocol (Secure) zugegriffen werden kann
- WWW wurde Anfang der 90er von Robert Cailliau und Tim Berners-Lee im Kernforschungszentrum CERN eingeführt
- Nutzer greifen mit Hilfe eines **Browsers** auf das Datenlager des WWW zu
- Browser (WWW-Client) bieten heute eine intuitive grafische Benutzeroberfläche – **GUI, Graphical User Interface**
- Weit verbreitete Browser sind: Chrome, Edge, Safari, Firefox, Opera, (IE)
- Browser interpretiert angeforderte WWW-Dokumente, bereitet sie auf und bringt sie zur Darstellung (Texte, Grafiken, Bilder, Videos, Audio, …)

How We Made the Internet

- https://www.youtube.com/watch?v=VPToE8vwKew

WWW-Dokumente

- Die im WWW angebotenen Informationen liegen in Form von **Hypermedia-Dokumenten** (auch **Hypertext-Dokumenten**) vor
- Hypermedia-Dokumente sind untereinander durch **Hyperlinks** (kurz: **Links**) verbunden und bilden auf diese Weise ein Netzwerk von Informationen und Ressourcen
- Die Startseite für die Navigation durch das Informationsangebot eines Anbieters heißt **Homepage**
- Die ggf. durch einen direkten Link zuerst aufgerufene Seite des Informationsangebotes eines Anbieters heißt **Landing Page**

WWW-Dokumente

- Um angefordertes und über das Internet angeliefertes WWW-Dokument richtig darstellen zu können, muss der Browser über die Vorstellungen des Dokumentenautors in Kenntnis gesetzt werden
- Um hier die notwendige Flexibilität zu erlangen, wird zwischen Dokumentenstruktur und Dokumentendarstellung unterschieden
  - **Dokumentenstruktur** beschreibt Aufbau und Untergliederung eines Dokuments, also Überschriften, Kapitel, Gliederungen, Absätze, Tabellen, …
  - **Dokumentendarstellung** betrifft Formatierung- und Layout des Dokuments

Strukturbeschreibung mit HTML

- Die Beschreibung der Struktur von WWW-Dokumenten erfolgt mittels der **Hypertext Markup Language - HTML**
- HTML ist eine sogenannte Markup-Beschreibungssprache:
  - Die einzelnen Strukturelemente werden mit HTML-Kennzeichnungen, den sogenannten **Markups**, ausgezeichnet
  - Die Markup-Kennzeichnungselemente sind sogenannte **Tags**
- **Browser** können das angeforderte HTML-Dokumente auf der Basis der HTML-Markups die Struktur des Dokuments erkennen (interpretieren)

Strukturbeschreibung mit HTML

- Jedes HTML-Dokument besteht aus zwei Teilen:
  - **„Header“** – enthält Informationen über das Dokument
  - **„Body“** – enthält eigentlichen Inhalt des Dokuments
- Mit Hilfe spezieller Tags können in einem HTML-Dokument auch Links als passive Zeiger auf andere Dokumente gespeichert werden

Layoutbeschreibung mit CSS

- Während HTML die Dokumentstruktur beschreibt, sind die **Cascading Stylesheets – CSS** – zuständig für Beschreibung der Darstellung (Layout) dieser Struktur
- Mittels CSS ist es möglich, das
  - Layout der einzelnen Strukturelemente eines HTML-Dokuments zu definieren,
  - Layout an verschiedene Ausgabemedien anzupassen, großer/kleiner Bildschirm, akustische Ausgabe, …
  - Layout des Informationsangebots zentral zu managen

Identifikation mit URIs

- Zur weltweit eindeutigen Identifikation und Adressierung von WWW-Dokumenten dienen sogenannte **URIs – Uniform Resource Identifier**
- Verbreitetste Ausprägung der URIs sind **URLs – Uniform Resource Locator** –, z.B. https://www.salzburgresearch.at/event/european-researchers-night-exploresearch-in-salzburg/
- Bestandteile einer URL sind:
  - Name des Zugriffsprotokolls, im Beispiel: https
  - Adresse des Servers, im Beispiel: salzburgresearch.at
  - Name des Dokuments, im Beispiel: /event/european-researchers-night-exploresearch-in-salzburg/

Kommunikation zwischen Browser und Server

- Zugriff auf WWW-Dokumente ist nach Client/Server-Paradigma organisiert:

1. Benutzer fordert über den Browser (WWW-Client) ein WWW-Dokument an (Maus-Klick, Eingabe einer URL, …)
2. Browser kontaktiert den in der URL spezifizierten WWW-Server und fordert gewünschtes Dokument an
3. Server greift auf sein lokales Filesystem zu und sendet die in der URL spezifizierte Datei zum anfragenden Browser
4. Browser empfängt Dokument
5. Browser interpretiert das Dokument und stellt es dar – analysiert HTML-Struktur des Dokuments und wendet auf Strukturelemente entsprechende CSS-Anweisungen an

Kommunikation zwischen Browser und Server

- Interaktion zwischen Browser und Server erfolgt mittels des **Hypertext Transfer Protocol - HTTP**
- HTTP ist sehr **einfaches und zustandsloses**, darum schnelles Protokoll – Interaktion erfolgt lediglich in Form eines einfachen Frage/Antwort-Verfahrens
- HTTP unterliegt stetem Entwicklungsprozess
  - erste HTTP-Version (HTTP/0.9) entstand 1989/90 am CERN
  - derzeit ist größtenteils noch Version HTTP/1.1 im Einsatz
  - aktuell: neue Version HTTP/2.0 ist fertig und wird von den meisten aktuellen Browsern bereits unterstützt

Identifikation von Ressourcen im WWW

- Grundbausteine des **WWW** – World Wide Web – sind **Hypermedia-Dokumente** mit ihren Verknüpfungen, den **Hyperlinks**
- Hypermedia-Dokumente sind weltweit verteilt auf Web-Servern gespeichert. Zur Lokalisierung und korrekten Verknüpfung wird eindeutiges Identifikationsschema benötigt
- Nutzen im täglichen Leben zwei Identifikationsmodelle:
  - Identifikation über den **Namen**
    - ist lebenslang gültig, oft aber nicht eindeutig
  - Identifikation über die **Anschrift**
    - hierarchischer Aufbau: Land, Stadt, Straße, Nummer, Wohneinheit
    - eindeutig bestimmt, aber ändert sich gelegentlich

Identifikation von Ressourcen im WWW

- Identifikationsschema im WWW: **URI - Uniform Resource Identifier**
- URIs haben zwei Ausprägungen:
  - **Uniform Resource Name – URN**:
    - Namensbeschreibung für Web-Ressource
  - **Uniform Resource Locator – URL**:
    - Adressbeschreibung für Web-Ressourcen
- In der Praxis des WWW haben sich bisher nur URLs durchgesetzt
  - **Vorteil**: Eindeutige Identifikation der Ressourcen leicht möglich
  - **Nachteil**: Adressänderungen nicht automatisch nachvollziehbar
- URNs führen Nischendasein, es fehlt Infrastruktur für die Auflösung von URNs (wie etwa DNS für URLs). ISBN, DOI, und NBN (National Bibliography Numbers) können als URNs aufgefasst werden

URI – Uniform Resource Identifier

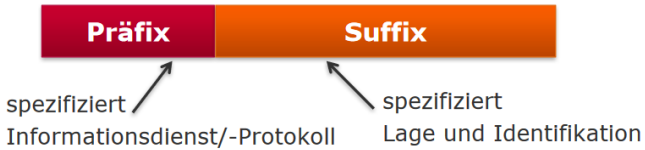
Anforderungen an Uniform Resource Identifier

- **Universalität** – über URI soll jede im Internet verfügbare Ressource ansprechbar sein, unabhängig von ihrem jeweiligen Informationsdienst
- **Eindeutigkeit** – jede Ressource muss weltweit eindeutig identifizierbar sein
- **Erweiterbarkeit** – auch Ressourcen, die über neue Informationsdienste angeboten werden, müssen zugreifbar sein
- **Fixierbarkeit** – URI sollen nicht nur auf elektronischem Weg austauschbar, sondern auch manuell bearbeitbar oder druckbar sein

URI – Uniform Resource Identifier

URI-Syntax

- URI wurden von IETF und W3C in RFC 1630 standardisiert
- Generelle Gestalt einer URI: **Präfix „:“ Suffix**



- RFC 1630 spezifiziert neben vollständigen URIs auch relative URIs
- Nach RFC 1630 ist URI entweder eine
  - **URL** – spezifiziert Lokalität einer Ressource – oder ein
  - **URN** – spezifiziert Name einer Ressource

URL – Uniform Resource Locator

URL gibt die genaue Adresse des Servers an, auf dem eine bestimmte Informationsressource zu einem Zeitpunkt gespeichert ist

- URL-Syntax (RFC 1738, 1808) folgt URI-Syntax: **Präfix : Suffix**
  - Präfix – Schema
  - Suffix – schemenspezifischer Teil und Parameter

URL – Uniform Resource Locator

Mögliche Schemata nach RFC 1738:

- ftp, z.B. ftp://speedtest.tele2.net
- http, z.B. http://localhost:8080/Conference-Portal
- https, z.B. https://salzburgresearch.at
- rtsp, z.B. rtsp://webradio.com/stream (Real-Time Streaming Protocol)
- mailto – Link zu einer Emailadresse
- news
- nntp – Usenet News über NNTP-Zugang
- telnet
- magnet – Ressourcen in Peer-To-Peer-Netzwerken
- file – Host-spezifische Dateinamen

URL – Uniform Resource Locator

Schemenspezifischer Teil:

„//“[benutzer[„:“passwort]„@“]host[„:“port]„/“pfad

(eckige Klammern kennzeichnen optionalen Eintrag)

- **Benutzer** – sinnvoll nur bei Zugriffsbeschränkungen auf die Ressource
- **Passwort** – zur Authentifikation eines Benutzers
- **Hostname** – vollständig qualifizierender Name oder IP-Adresse
- **Portnummer** – Verbindungsport für aufzubauende Verbindung; ist bei meisten Diensten bereits festgelegt
- **Pfadname** – spezifiziert, wie auf angegebenen Host mit angegebenem Dienst auf die Ressource zugegriffen werden kann

URN – Uniform Resource Name

- URN dient der weltweit eindeutigen und dauerhaften Identifikation einer Informationsressource über einen Namen
- Um zu entscheiden, ob URI eine URL oder URN festlegt, muss Präfix ausgewertet werden
- Zur Zeit werden URNs kaum unterstützt. Eines der wenigen Beispiele: Deutsche Nationalbibliothek => http://nbn-resolving.de/
- URN-Syntax wurde in RFC 1630/RFC 2141 definiert, funktionale Eigenschaften in RFC 1737
- Liste mit URN-Namespaces einsehbar unter: http://www.iana.org/assignments/urn-namespaces

HTTP - Einführung

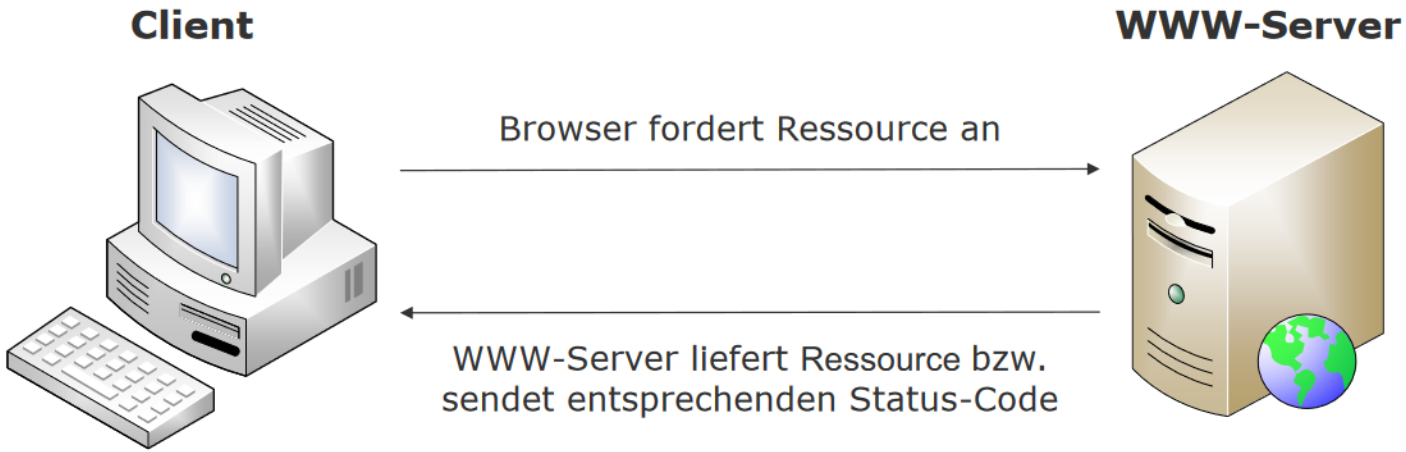
- WWW – World Wide Web – ist gigantisches Hypermediasystem mit weltweit verteilten Ressourcen
- Zugriff auf die Ressourcen des WWW wird durch das HTTP-Protokoll geregelt, das Prozeduren zum Abruf und zur Anlieferung der durch URIs eindeutig gekennzeichneten Ressourcen bereitstellt
- **HTTP – Hypertext Transfer Protocol** – regelt Kommunikation zwischen dem informationsanfordernden **WWW-Client – Browser** und einem informationsanbietenden **WWW-Server**
- Nutzer kommt selbst nicht mit HTTP in Berührung – Browser veranlasst angestoßen durch entsprechende Nutzeraktionen auf grafischer Benutzeroberfläche seines Browsers die jeweils notwendige Abfolge von HTTP-Befehlen und arbeitet diese ab

HTTP - Basisoperationen

- HTTP setzt auf zuverlässigem, verbindungsbasierten Transportdienst TCP auf
- **Ablauf:**
  - Browser initiiert als Client eine Kommunikation durch Anforderung einer Informationsressource bei einem WWW-Server (**Request**)
  - WWW-Server nimmt Anforderung entgegen, verarbeitet sie und antwortet entsprechend (**Response**)
    - Ist Ressource verfügbar und darf Browser auf sie zugreifen, sendet der Server die Ressource zusammen mit einem positivem Statuscode
    - Ist Ressource nicht verfügbar oder ist Zugriff für Browser verboten, sendet der Server einen negativen Statuscode
- HTTP ist ein **zustandsloses Protokoll**, d.h. besitzt keine Kenntnis von bereits erfolgten Anfragen-Antworten-Zyklen

HTTP - Basisoperationen

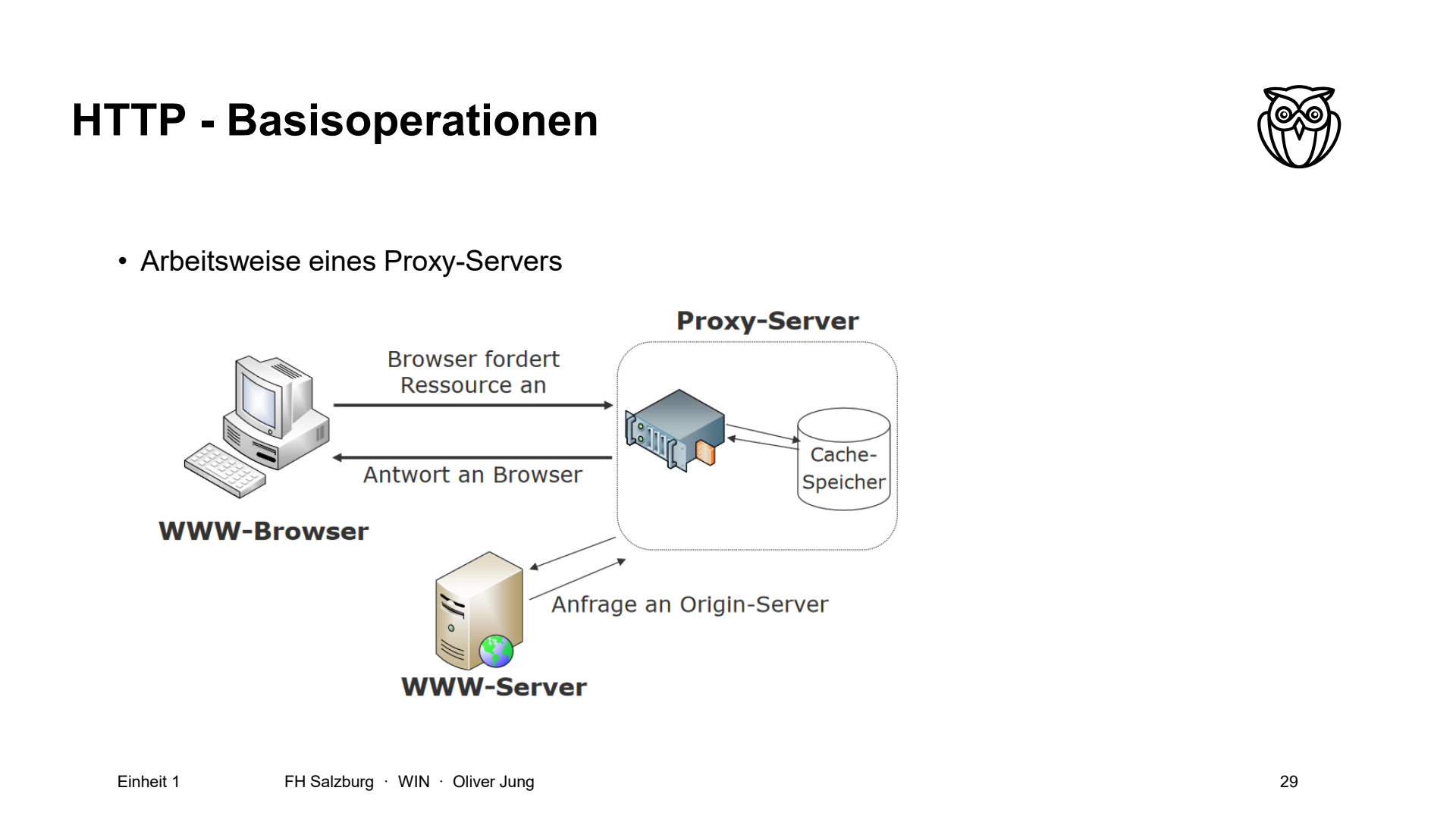
- HTTP Client-Server Architektur



HTTP - Basisoperationen

- In der Praxis ist Interaktion zwischen Browser und Server komplexer, da verschiedene Zwischensysteme – Proxy-Server und Gateways – in die Kommunikation eingebunden sind
- **Proxy-Server**
  - Zwitterstellung in der Kommunikation zwischen Browser und Server
    - arbeitet gegenüber Client als Server, wenn er Anforderung aufgrund einer früheren Kommunikation aus seinem Cache-Speicher bedienen kann
    - arbeitet gegenüber Server – **Origin Server** – als Client, indem er Client-Anforderungen, die er nicht bedienen kann, weiterleitet
  - Alle Browser-Anfragen können über einen Proxy-Server weitergeleitet werden (einstellbar)

HTTP - Basisoperationen



- Arbeitsweise eines Proxy-Servers

HTTP - Basisoperationen

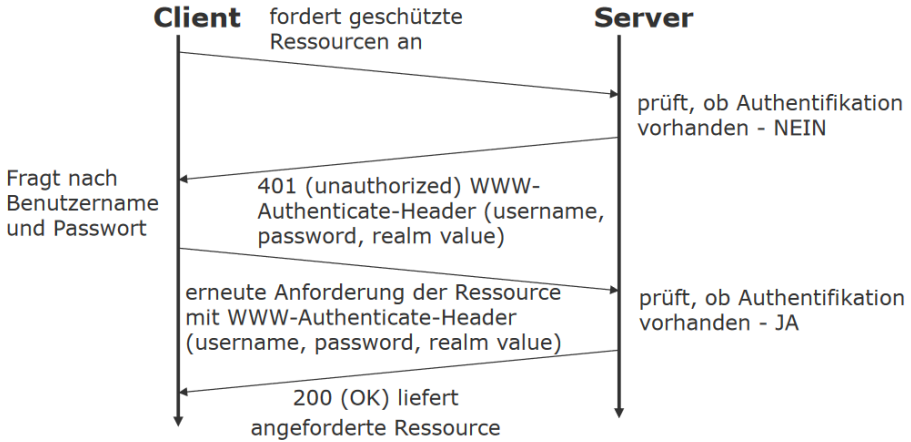
- **Gateways**
  - Arbeiten wie Proxy-Server nur ohne Kenntnis des Clients
  - Gateways werden WWW-Servern vorgeschaltet, um diese zu entlasten oder sicherheitsbedingte Zugriffsrestriktionen zu implementieren
  - Auch **Load Balancer** sind Beispiel für HTTP Gateways

HTTP - Authentifikation

- Soll auf sicherheitsrelevante Daten des WWW-Servers zugegriffen werden, ist korrekte Authentifikation und Autorisierung des Clients erforderlich
- HTTP muss Authentifikation und Autorisation mit zustandslosen Methoden regeln:
  - Client sendet Anforderung für geschützte Ressource
  - Server prüft Verfügbarkeit der Ressource und antwortet mit Status Code **401 Unauthorized** zusammen mit **WWW-Authenticate** Response-Headerfeld
  - Client sendet neuen Request mit Authorization-Headerfeld, das die vom Server im Authenticate-Response-Headerfeld angeforderten Credentials in der verlangten Form beinhaltet
- HTTP/1.1 verwendet zwei verschiedene Methoden zur Authentifikation: **Basic Authentication** und **Digest Access Authentication**

HTTP - Authentifikation

- Basic Authentication



- **Problem:** Benutzername und Passwort werden bei unverschlüsselter Verbindung im Klartext übertragen!

HTTP - Authentifikation

Sichere Alternative: HTTPS (RFC 2818)

- Hypertext Transfer Protocol Secure
- **Verschlüsselung und Authentifizierung** der Kommunikation zwischen Webserver und Browser (Client)
- Prinzip ist einfach: HTTP über TLS
- Realisierung über zusätzliche (Teil-)Schicht im TCP/IP-Stack
- TLS (Transport Layer Security), früher bekannt als SSL, sorgt für Verschlüsselung
- Passwörter und andere vertrauliche Daten können bei HTTPS nicht mehr ohne weiteres von unberechtigten Dritten mitgelesen werden

HTTP - Methoden

- GET – Anforderung eines Dokuments vom Server durch den Client
- POST – Informationsübertragung vom Client zum Server
- HEAD – ähnlich GET, aber nur Nachrichtenkopf wird angefordert
- PUT – Hinzufügen einer Ressource auf dem Server
- PATCH – Aktualisieren eines Dokuments, ohne es ganz zu ersetzen (PUT)
- DELETE – Löschen einer Ressource vom Server
- OPTIONS – Anfordern der vom Server unterstützten HTTP-Methoden
- TRACE – Auslieferung der vom Server tatsächlich empfangenen Anfrage (z.B. zur Fehlersuche bei der Entwicklung)
- CONNECT – Aufbau eines verschlüsselten Tunnels

HTTP - Nachrichtenformat

Alle HTTP-Nachrichten folgen einheitlicher Struktur, dem Generic Message Format

- **Startsequenz** legt fest, ob Nachricht eine Anfrage oder Antwort ist und enthält entspr. Informationen
- **Nachrichtenkopf** besteht aus Name-/Wert-Paar welches mit Doppelpunkten getrennt ist, z.B. `<header-name>: <header-value>`
- **Leerzeile** zur Trennung von Kopf und Rumpf
- **Nachrichtenrumpf** enthält die zu übertragenen Informationen, z.B. die angeforderten Ressourcen oder detaillierte Fehlermeldungen
  - ist optional, da nicht alle HTTP-Nachrichten einen Nachrichtenrumpf benötigen

HTTP – Request-Nachrichtenformat

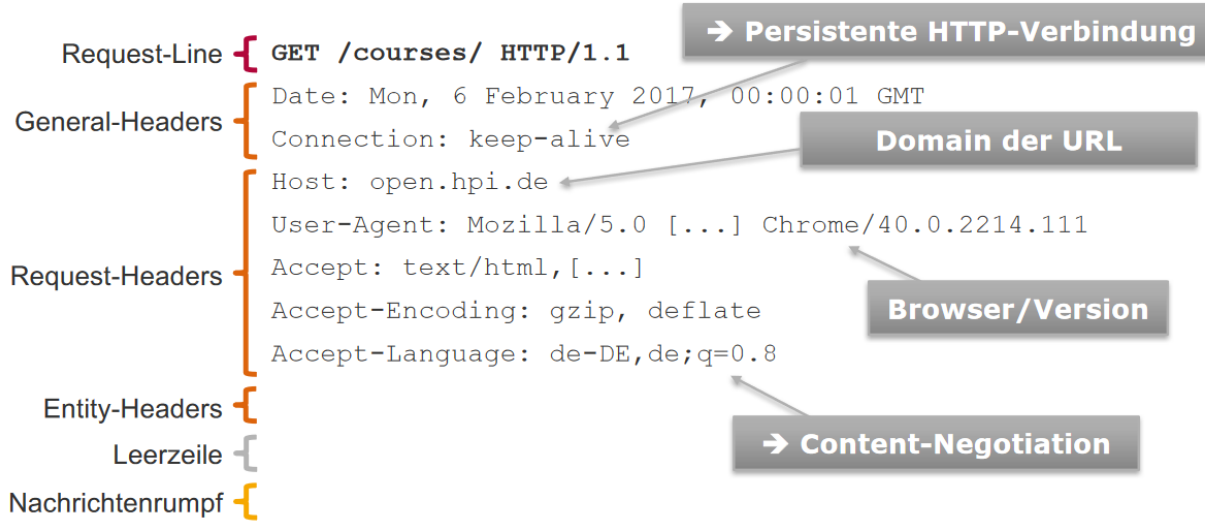
Client initiiert **HTTP-Session**, indem eine TCP-Verbindung zum Server aufgebaut und eine HTTP-Request-Nachricht gesendet wird

Request Nachrichtenformat:

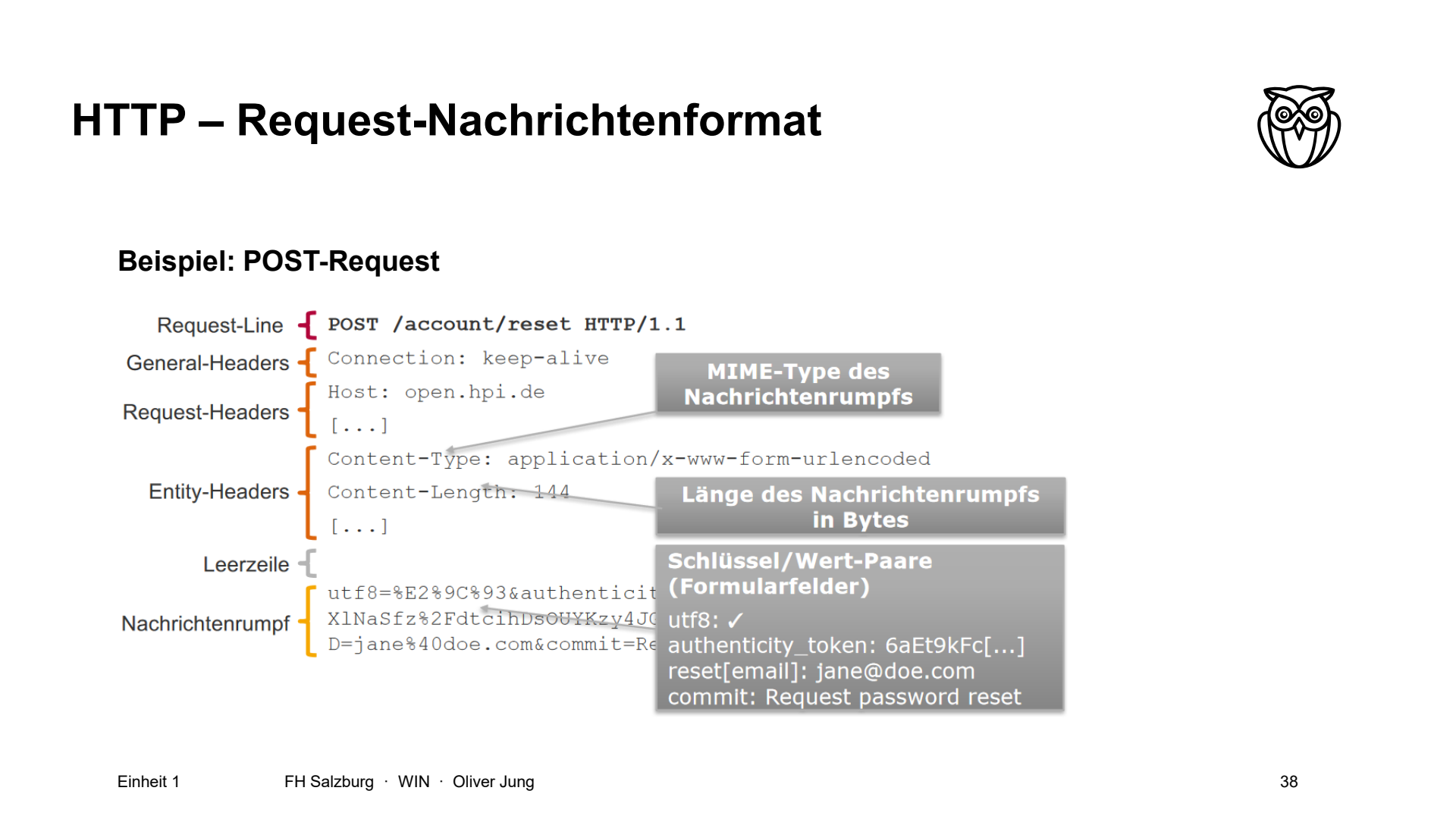
- **Startsequenz** heißt **Request-Line** und enthält Methode, Request-URI und HTTP-Version, z.B. *GET /courses HTTP/1.1*
  - URLs der Request-Line sind typischerweise relativ
- **Nachrichtenkopf** besteht aus diversen Header-Typen
  - **General-Header** enthält Informationen zur Nachricht (nicht anfrage- oder antwortspezifisch)
  - **Request-Header** enthält Details zur Anfrage
  - **Entity-Header** beschreibt Nachrichteninhalt (wenn vorhanden)

HTTP – Request-Nachrichtenformat

Beispiel: GET-Request



HTTP – Request-Nachrichtenformat



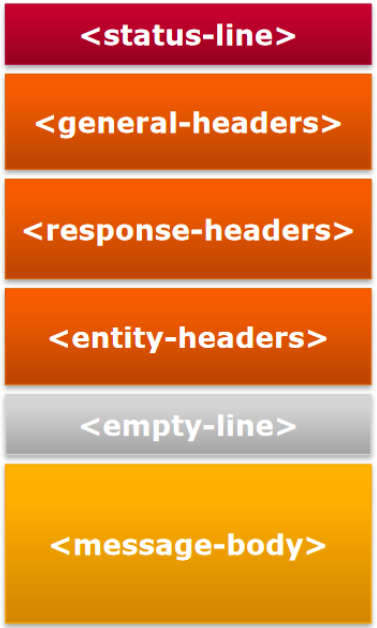
Beispiel: POST-Request

HTTP – Response-Nachrichtenformat

Server liefert Ressource oder Fehlermeldung als **Response**-Nachricht aus

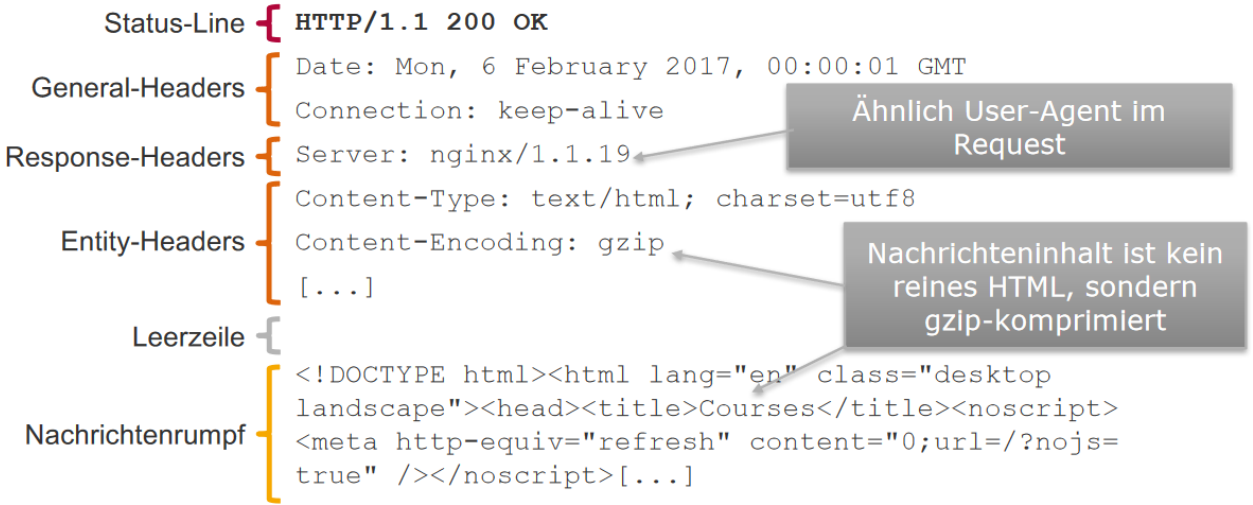
Response-Nachrichtenformat:

- **Startsequenz** einer Antwort ist **Status-Line**: HTTP-Version, Status-Code, Status-Nachricht, z.B. `HTTP/1.1 404 Not found`
  - HTTP-Version sollte nicht höher sein, als die Version der zugehörigen Anfrage
- **General-Header** und **Entity-Header** sind identisch wie bei den Request-Nachrichten
- **Response-Headers** enthalten zusätzliche Informationen für Client, die in Abhängigkeit vom Statuscodes variieren
- **Nachrichtenrumpf** enthält übertragene Ressource oder Fehlerinformationen



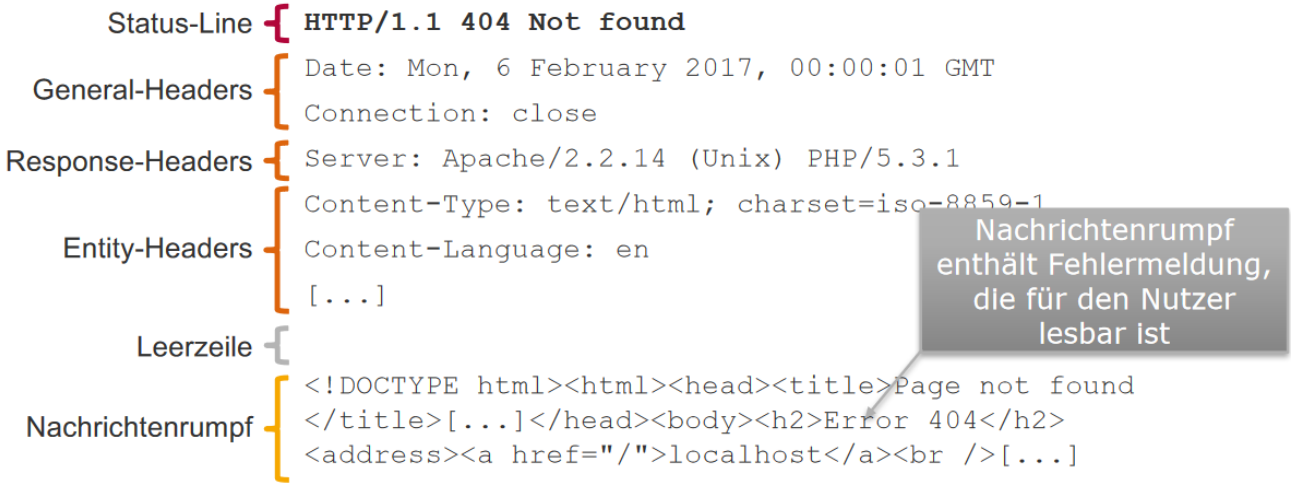
HTTP – Response-Nachrichtenformat

Beispiel: Erfolgreicher Response



HTTP – Response-Nachrichtenformat

Beispiel: Angeforderte Ressource nicht gefunden



HTTP – Response-Nachrichtenformat

Beispiel: Angeforderte Ressource wurde verschoben

Status-Line
```
HTTP/1.1 301 Moved Permanently
```
General-Headers
```
Date: Mon, 6 February 2017, 00:00:01 GMT
Connection: close
```
Response-Headers
```
Server: Apache/2.2.14 (Unix) PHP/5.3.1
Location: http://www.new-website.com
```
Entity-Headers …

Leerzeile

Nachrichtenrumpf …

Response-Header `location` enthält URL des neuen Orts der Ressource; der Browser fordert dann diese Ressource automatisch von dort an

HTTP - Statuscodes

HTTP definiert verschiedene Statuscode-Klassen

- **1xx – Informative Nachrichten**
  - vorläufige Antwort zu Informationszwecken
  - Beispiel: **101 Switching Protocols** – Client erbittet Änderung des Protokolls, Server bestätigt die Änderung
- **2xx – Success**
  - Codes zeigen, dass der Server die Anfrage erfolgreich empfangen und verarbeitet hat
  - Beispiel: **200 OK**
- **3xx – Umleitung**
  - Codes zeigen an, dass der Client eine zusätzliche Aktion auszuführen hat (z.B. zusätzlicher Request), um Ressource zu erhalten
  - Beispiel: **301 Moved Permanently**

HTTP - Statuscodes

- **4xx – Client-Fehler**
  - Codes zeigen fehlerhaften Client-Request an oder angeforderte Ressource existiert nicht oder höhere Berechtigung wird benötigt
  - Beispiel: **400 Bad Request** – Client-Request war syntaktisch inkorrekt
- **5xx – Server-Fehler**
  - Anfrage des Clients ist valide, aber Server ist nicht in der Lage, die Anfrage zu verarbeiten
  - dem Server ist bewusst, dass Fehler auf Serverseite liegt
  - Beispiel: **503 Service Unavailable** – Server kann (derzeit) die Anfrage nicht beantworten, z.B. wegen Überlast

Offizieller Datenbestand der HTTP Status-Codes wird von IANA gepflegt:
http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

HTTP - Optimierung

- Moderne Webseiten enthalten zahlreiche **eingebettete Ressourcen** wie z.B. Stylesheets, JavaScript, Bilder, Videos, …
- Bei Ausführung eines HTTP-Request/-Response-Zyklus muss zunächst TCP-Verbindung auf- und dann wieder abgebaut werden
  - Auf- und Abbau einer TCP-Verbindung verlangt Ausführung eines TCP-Handshakes und verursacht dadurch signifikanten Overhead
- Es werden dringend HTTP-Mechanismen gebraucht zur Request-Minimierung und zur Geschwindigkeitserhöhung, wie
  - **Persistente Verbindungen** („Connection: keep-alive“ header) und **HTTP-Pipelining** (Anfrage-Sequenzen parallelisieren)
  - **Komprimierung** (Nachrichtenrumpf komprimiert übertragen; gzip, deflate, …)
  - **Caching** (clientseitig, eigenständig, serverseitig)
    - Wichtig: Cache-Konsistenz muss sichergestellt werden (Gültigkeitsdauer)

HTTP - Cookies

- **Cookies** bieten Konzept zur Bildung von **Sessions** / Übertragung von **Session-Ids** (Zusammenfassung zusammengehörender Nutzerinteraktionen)
  - wurden in den 90er-Jahren von Netscape entwickelt
  - Cookie-Mechanismus erlaubt Austausch von Zustandsinformationen, ohne dass dazu persistente Netzwerkverbindungen nötig sind
- Anwendungsfälle:
  - Nutzer-Präferenzen speichern, z.B. bevorzugte Sprache
  - Identifikation der Nutzer
  - Tracking (z.B. Werbe-Netzwerke), oft gemeinsam mit weiteren Erkennungsmerkmalen (z.B. Version von Browser und Betriebssystem)

HTTP - Cookies

Ablauf

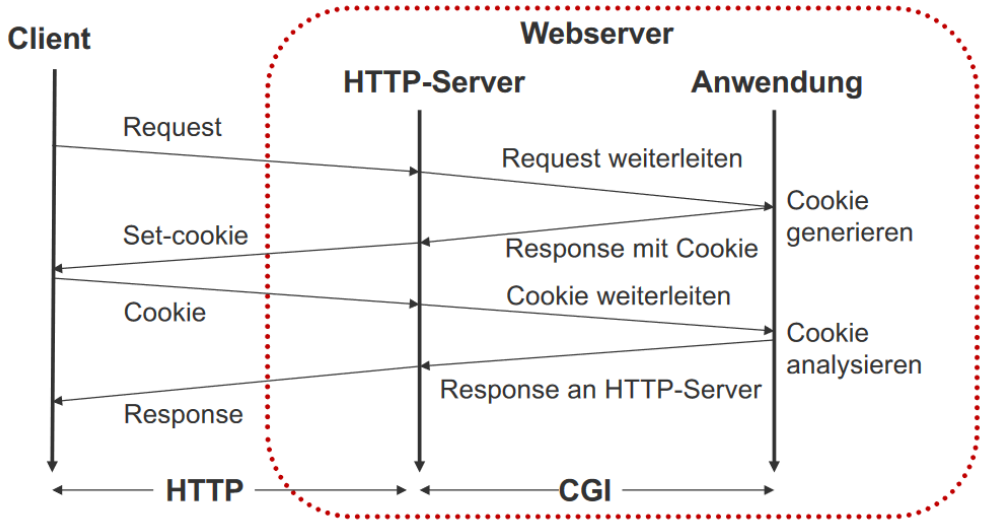
1. HTTP-Server beauftragt den Client, Cookie anzulegen, indem er in einer HTTP-Antwort das Headerfeld set-cookie setzt
2. Browser speichert Cookie in einer internen Datenbank
3. Bei jedem Request an die gleiche Domain sendet Browser all seine dafür passenden Cookies im Request-Headerfield cookie

Cookies haben begrenzte Größe

- Meist werden nur (Session) IDs in Cookies gesendet. Eigentliche Informationen (z.B. virtueller Warenkorb) werden serverseitig abgelegt

HTTP - Cookies

Cookie-Mechanismus



HTTP - Cookies

Vorteile

- Bei Übertragung mit HTTPS bleiben Session-Informationen geheim
- Einfachere Wartbarkeit für Webseitenbetreiber und –Entwickler
- Session-Handling unabhängig von HTML

Nachteile

- Session-Informationen aus HTTP-Headern auslesbar, wenn diese unverschlüsselt übertragen wird
- Tracking von Nutzer-Interaktionen sehr einfach möglich für Webseitenbetreiber und Dritte (Google, Facebook, …)

FH Salzburg

VO Web-Technologien

Einheit 1, Oliver Jung

Technik
Gesundheit
Medien