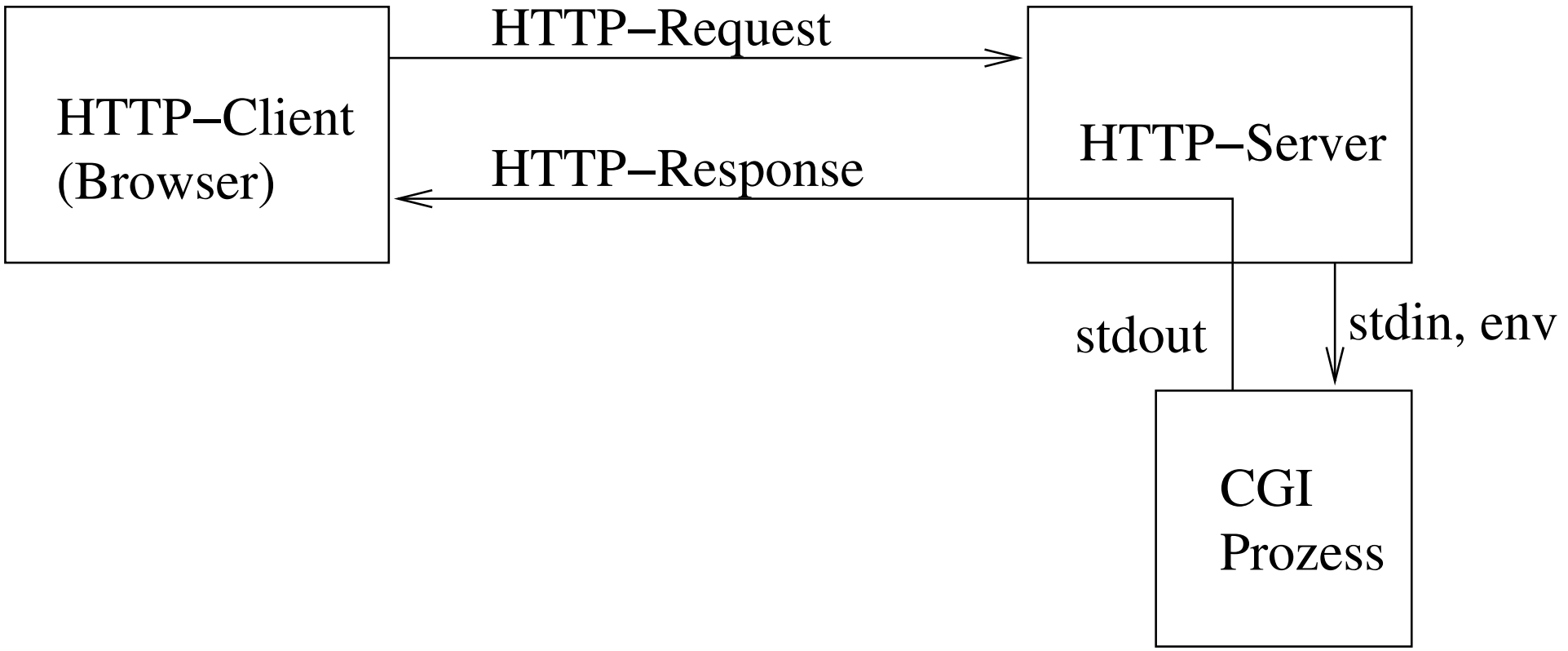
9 Dynamisches Erzeugen von Webseiten

9.1 CGI

CGI (Common Gateway Interface) Skripte erlauben die dynamische Erzeugung von Dokumenten auf dem Webserver.

Typische Anwendung: CGI-Skripte verarbeiten Eingaben aus Formularen und erzeugen in Abhängigkeit von den Eingaben ein Antwortdokument.

Überblick



Eigenschaften von CGI

Einfachheit

Sprachunabhängigkeit

Sicherheit durch separaten Prozess

Offener Standard

Architekturunabhängigkeit

Ausführung eines CGI-Skripts

- Server erkennt CGI-Skripte an der URL
  - spezielles Element im Pfad (z.B. `cgi-bin`), dann nächstes Pfadelement
    = Name eines ausführbaren Programms in konfigurierbarem Verzeichnis
  - spezielle Extension am Dateinamen (z.B. `.cgi`)
    = Name eines ausführbaren Programms
- Server verarbeitet den Header des HTTP-Requests
- Legt Request-Information in Environment ab (Prozessumgebung)
- Generiert die Statuszeile und einige Response-Header (`Date`, `Server`, `Connection`)
- Schliesst den Headerteil der Response **NICHT** ab
- Startet das CGI-Programm mit
  - Standardausgabe $\Leftrightarrow$ Versenden an Client/Browser
  - Standardeingabe $\Leftrightarrow$ ggf. Lesen vom Client/Browser
  - Argumente $\Leftrightarrow$ Pfadelemente **nach** dem Namen des CGI-Programms
  - Umgebung definiert weitere Parameter der Anfrage

Pflichten eines CGI-Programms

- Interpretation der Parameter und der Anfrage
- Drucken weiterer Headerzeilen
  (`Content-Length`, `Content-Type`, `Content-Encoding`, ...)
- **Abschliessen des Headerteils der Response durch Leerzeile**
- Generieren des Inhaltes entsprechend `Content-Type`

Sprachen zur CGI-Programmierung

- Jede Sprache geeignet, die Standardeingabe und Umgebungsvariable lesen kann, sowie Standardausgabe schreiben kann
- Für Java ist ein *wrapper* Programm zum Lesen der Umgebungsvariablen erforderlich
- Manche Webserver beinhalten Interpreter für Skriptsprachen (perl, php, etc), um die Startzeit für einen externen Interpreter zu sparen
  Beispiel: Apache Module mod_perl, mod_php, mod_python, mod_ruby, ...

9.1.1 Parameter für ein CGI-Programm

Die Einsendung eines XHTML Formulars liefert

$Feldname_1$=$Wert_1$
$Feldname_2$=$Wert_2$
...
$Feldname_k$=$Wert_k$

wobei Feldnamen wiederholt auftreten können.

Feldnamen und Werte werden vor Übertragung vom Browser kodiert

Standardkodierung: URL Kodierung `application/x-www-form-urlencoded`

- Buchstaben und Zahlen bleiben erhalten
- Leerzeichen werden durch + ersetzt
- Alle weiteren Zeichen werden durch %⟨*ASCII-code*⟩ ersetzt
  (in zweistelliger Hexadezimaldarstellung)

vgl.

```
public static String java.net.URLEncoder.encode(String s)
```


9.1.2 Zugriffsmethoden

`GET` Kodierung der Anfrage in der URL durch Anhängen eines ⟨*Querystring*⟩ der Form `?`⟨*Feld-Wert-Liste*⟩ an die `action` URL

$$\langle \textit{Feld-Wert-Liste} \rangle \quad ::= \quad \langle \textit{kodierter-Feldname} \rangle = \langle \textit{kodierter-Wert} \rangle \left(\& \langle \textit{kodierter-Feldname} \rangle = \langle \textit{kodierter-Wert} \rangle \right)^*$$

Der Webserver legt den ⟨*Querystring*⟩ in der Umgebungsvariable `QUERY_STRING` ab.

`POST` verschickt die ⟨*Feld-Wert-Liste*⟩ im ⟨*message-body*⟩ der Anfrage. Der Webserver speichert die Länge (in Octets) des ⟨*message-body*⟩ in der Umgebungsvariable `CONTENT_LENGTH`. Das CGI-Programm muss **genau** so viele Octets lesen und interpretieren (nicht bis Dateiende lesen!)

Rationale für Methoden: `GET` beobachtet, `POST` ändert ggf. Zustand des Servers

Kodierung und Zeichensatz

Durch URL Kodierung werden

- unerlaubte Zeichen in URLs vermieden
- die Zeichen = und & in Feldnamen und Werten verwendbar

Problem: Zeichensatz-Kodierung

- Viele Möglichkeiten: US-ASCII, iso-8859-1, utf-8, utf-16, iso-2022-jp, EUC-JP, windows-1252
- Siehe `http://www.iana.org/assignments/character-sets`

Welche Zeichensatz-Kodierung wird für Requestdaten verwendet?

- Zeichensatz des Formulars
- Vorgabe durch `accept-charset` Attribut des `<form>`
- Erst Zeichensatz-Kodierung, dann URL Kodierung

Beispiele

GET Suche nach würg (iso-8859-1, utf-8)

```
http://www.google.com/search?q=w%FCrg&btnG=Google+Search
http://www.google.com/search?hl=en&lr=&q=w%C3%BCrg&btnG=Search
```

POST Klick auf Termine

```
s12x0=Termine&%3DCGI%3Dparm%3D=WzpWWygiczExeDEiLCJUYWd1bmdzb3J0IiksKCJmMTF4NyIsI
iIpLCgiZjExeDgiLCIiKSwoImYxMXg5IiwiIiksKCJmMTF4MTAiLCIiKSwoImYxMXgxMSIsIiIpLCgiZ
jExeDEyIiwiIiksKCJmMTF4MTMiLCIiKSwoImYxMXgxNCIsIiIpLCgiZjExeDE1IiwibzEiKSwoImYxM
XgxNSIsIm8yIiksKCJmMTF4MTUiLCJvMyIpLCgiZjExeDE1IiwibzQiKSwoImYxMXgxNiIsIm8zIiksK
CJmMTF4MTciLCJvMSIpXSw6VlsoInMxMHgyIiwiQW5tZWxkdW5nIildLDpWWygiczl4MyIsIkFucmVpc
2UiKV0sOlZbKCJzOHg0IiwiVGVpbG5laG1lciIpXSw6VlsoInM3eDUiLCJQcm9ncmFtbSIpXSw6VlsoI
nM2eDYiLCJQcm9jZWVkaW5ncyIpXSw6VlsoInM1eDUiLCJQcm9ncmFtbSIpXSw6VlsoInM0eDQiLCJUZ
WlsbmVobWVyIildLDpWWygiczN4MyIsIkFucmVpc2UiKSwoImYzeDciLCIiKSwoImYzeDgiLCIiKSwoI
mYzeDkiLCIiKSwoImYzeDEwIiwiIiksKCJmM3gxMSIsIiIpLCgiZjN4MTIiLCIiKSwoImYzeDEzIiwiI
iksKCJmM3gxNCIsIiIpLCgiZjN4MTUiLCJvMSIpLCgiZjN4MTUiLCJvMiIpLCgiZjN4MTUiLCJvMyIpL
CgiZjN4MTUiLCJvNCIpLCgiZjN4MTYiLCJvMyIpLCgiZjN4MTciLCJvMSIpXSw6VlsoInMyeDIiLCJBb
m1lbGR1bmciKV0sOlZbKCJzMXgxIiwiVGFndW5nc29ydCIpXSw6VlsoInMweDAiLCJUZXJtaW5lIildX
Q%3D%3D
```

Ein Shellskript als CGI-Skript: `date`

Abspeichern als `date.cgi` oder als `.../cgi-bin/date`.

```
#!/bin/sh

echo Content-type: text/html
echo
cat <<EOF
<HTML><head><title>Date and Time</title></head>
<body> <h1>Date and Time</h1>
EOF
/bin/date
cat <<EOF
<p>
<a href="/index.html">Back home</a>
</body></html>
EOF
```

9.1.3 Umgebungsvariablen im CGI-Skript: `test-cgi`

```
#!/bin/sh
# disable filename globbing
set -f
echo Content-type: text/plain
echo
echo CGI/1.0 test script report:
echo
echo argc is $#. argv is "$*".
echo
echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = "$PATH_INFO"
echo PATH_TRANSLATED = "$PATH_TRANSLATED"
echo SCRIPT_NAME = "$SCRIPT_NAME"
```



```
echo QUERY_STRING = "$QUERY_STRING"
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo AUTH_TYPE = $AUTH_TYPE
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

Aufrufendes Formular

```
<form action="/cgi-bin/test-cgi/extra/parameters" method="get">
  <input type="text" name="text=1" value="value for text=1">
  <input type="text" name="text=2" value="nothing really+&!">
  <input type="submit">
</form>
```

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/1.3.9 (Unix)
SERVER_NAME = hanauma.informatik.uni-freiburg.de
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
PATH_INFO = extra/parameters
PATH_TRANSLATED = /usr/local/www/data/extra/parameters
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING = text%3D1=value+for+text%3D1&text%3D2=nothing+really%2B%26%21
REMOTE_HOST =
REMOTE_ADDR = 127.0.0.1
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

9.2 File Upload

Wert eines Eingabefeldes ist Inhalt einer Datei

Probleme:

- Dateigröße macht Anhängen an URL unmöglich
- Dateiinhalt überwiegend nicht-ASCII:
  URL Kodierung expandiert die Dateigröße um Faktor 3
- Übertragung von Metadaten, wie Dateiname, content type, etc

Lösung: Verwendung von (Attributen von `<form>`)

- `method="post"`
- `enctype="multipart/form-data"` (neuer MIME-Typ RFC 1521)

Beispiel

```
<form enctype="multipart/form-data" action="/cgi-bin/test-cgi" method="post">
   File to process: <input name="userfile1" type="file" />
   <input type="submit" value="Send File" />
</form>
```

Browser verschickt Anfrage mit Header

`Content-type: multipart/form-data; boundary=`⟨*boundary-string*⟩

⟨*boundary-string*⟩ so gewählt, dass er nicht in den Felddaten vorkommt

Der ⟨*message-body*⟩ wiederholt

`--`⟨*boundary-string*⟩
`Content-Disposition: form-data; name="`⟨*Feldname*⟩`"[; filename="`⟨*Dateiname*⟩`"]`
`Content-Type:` ⟨*content type*⟩
|
⟨*Inhalt des Feldes*⟩

und endet mit

`--`⟨*boundary-string*⟩`--`

Mehrere Selektionen in einem Feld (Feld Header)

⟨*boundary-string*⟩
`Content-Disposition: form-data; name="`⟨*Feldname*⟩`"`
`Content-Type: multipart/mixed; boundary=`⟨*interner boundary-string*⟩
|
⟨*Inhalt*⟩

Inhalt eines `multipart/mixed` **Feldes**

- Wiederholung von

  `--`⟨*interner boundary-string*⟩
  `Content-disposition: attachment; filename="file1.txt"`
  `Content-Type:` ⟨*content-type*⟩
  |
  ⟨*... Inhalt von file1.txt ...*⟩

- Beenden mit

  `--`⟨*interner boundary-string*⟩`--`

Kodierung von Nicht-ASCII Zeichen (RFC 1522) in Feldnamen

$$
\begin{array}{lcl}
\langle \textit{encoded-word} \rangle & ::= & \texttt{=?}\langle \textit{charset} \rangle\texttt{?}\langle \textit{encoding} \rangle\texttt{?}\langle \textit{encoded-text} \rangle\texttt{?=} \\
\langle \textit{charset} \rangle & ::= & \texttt{ISO-8859-1} \mid \ldots \\
\langle \textit{encoding} \rangle & ::= & \texttt{B} \mid \texttt{Q}
\end{array}
$$

- Falls $\langle \textit{encoding} \rangle = \texttt{B}$, so ist $\langle \textit{encoded-text} \rangle$ mit base64 (RFC 1521) kodiert.
- Falls $\langle \textit{encoding} \rangle = \texttt{Q}$, so ist $\langle \textit{encoded-text} \rangle$ mit `quoted-printable` (RFC 1521) kodiert.

9.3 Sitzungsmanagement

Problem:

- HTTP ist zustandslos,
- aber interaktive Skripte wollen Sitzungen implementieren

$\Rightarrow$ muss außerhalb von HTTP implementiert werden

Was ist eine Sitzung?

- Logisch zusammengehörige Folge von Formularen und Formulareinreichungen
- Nächstes Formular kann von **allen** vorher in der gleichen Sitzung besuchten Formularen abhängen
- Beispiel: Einkaufswagen, Webmail, ...

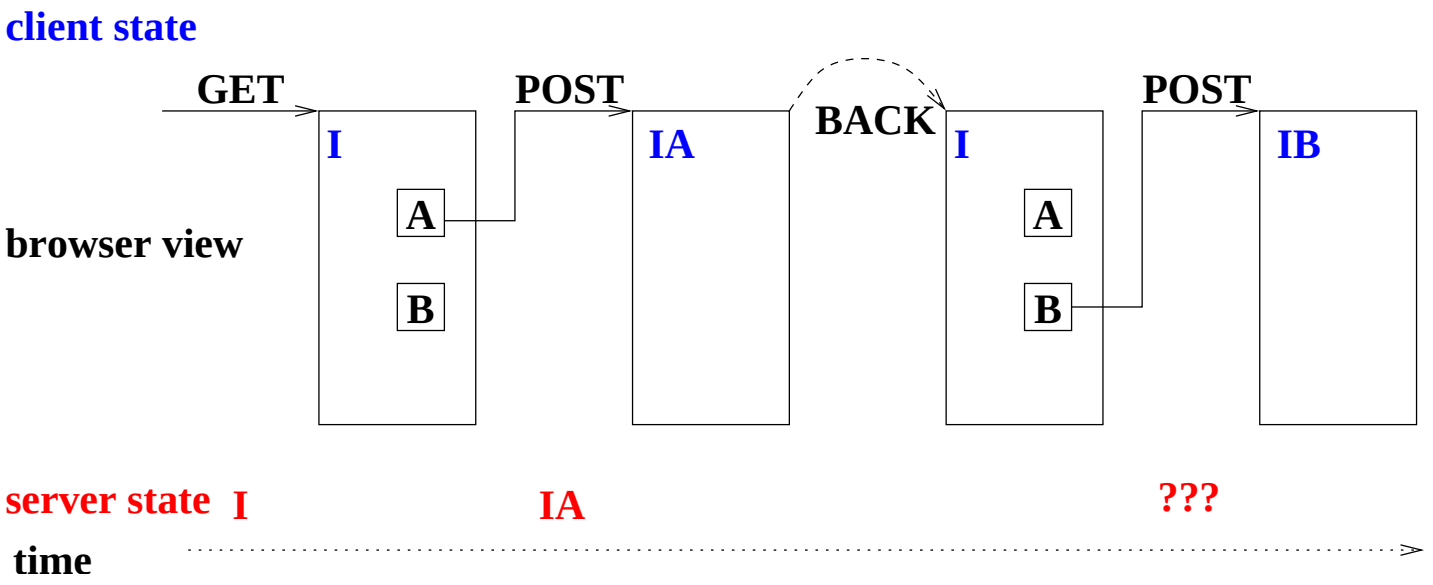
Ansätze zur Verwaltung des Sitzungszustands

Server generiert eine Sitzungs-ID und unterhält eine Abbildung:
Sitzungs-ID $\rightarrow$ Sitzungszustand

- Sitzungs-ID wird Teil aller Antworten / Formulare:
  1. als Teil der URL (URL rewriting)
  2. in versteckten Formularfeldern (hidden fields)
  3. in "cookies"

  (Servlets/JSP verwenden 3. bzw 1.)
- Problem bei Verwaltung des Sitzungszustands:
  Navigationsfunktionen des Browsers (back button)
- Timeout von Sitzungen erforderlich
  Abbildung wächst unbeschränkt

Problem: Browser Navigation und Serverzustand



- ähnliche Probleme mit Clonen und Bookmarks

Alternative Verwaltung des Sitzungszustands

Sitzungszustand in versteckten Formularfeldern
(WASH, BEA SOA)

+ kein Problem mit Browser Navigation
+ kein Sitzungszustand auf dem Server
  - $\Rightarrow$ kein Timeout für Sitzungen erforderlich
  - $\Rightarrow$ bessere Skalierbarkeit, da Server austauschbar
- Größe
- Integrität des Sitzungszustands
  (Sicherstellung durch kryptographische Verfahren)
- Synchronisation mit Serverzustand

– "`post`" erforderlich

9.3.1 Cookies

Problem HTTP zustandslos

Wunsch Persistenz

- Gruppierung von einzelnen Zugriffen zu einer *Sitzung* (Einkaufswagen, Ariadnefaden, etc)

  Eigenschaften: relativ kurze Lebensdauer, abbrechbar von beiden Seiten, implizit

- Profilinformation

  Eigenschaften: lange Lebensdauer

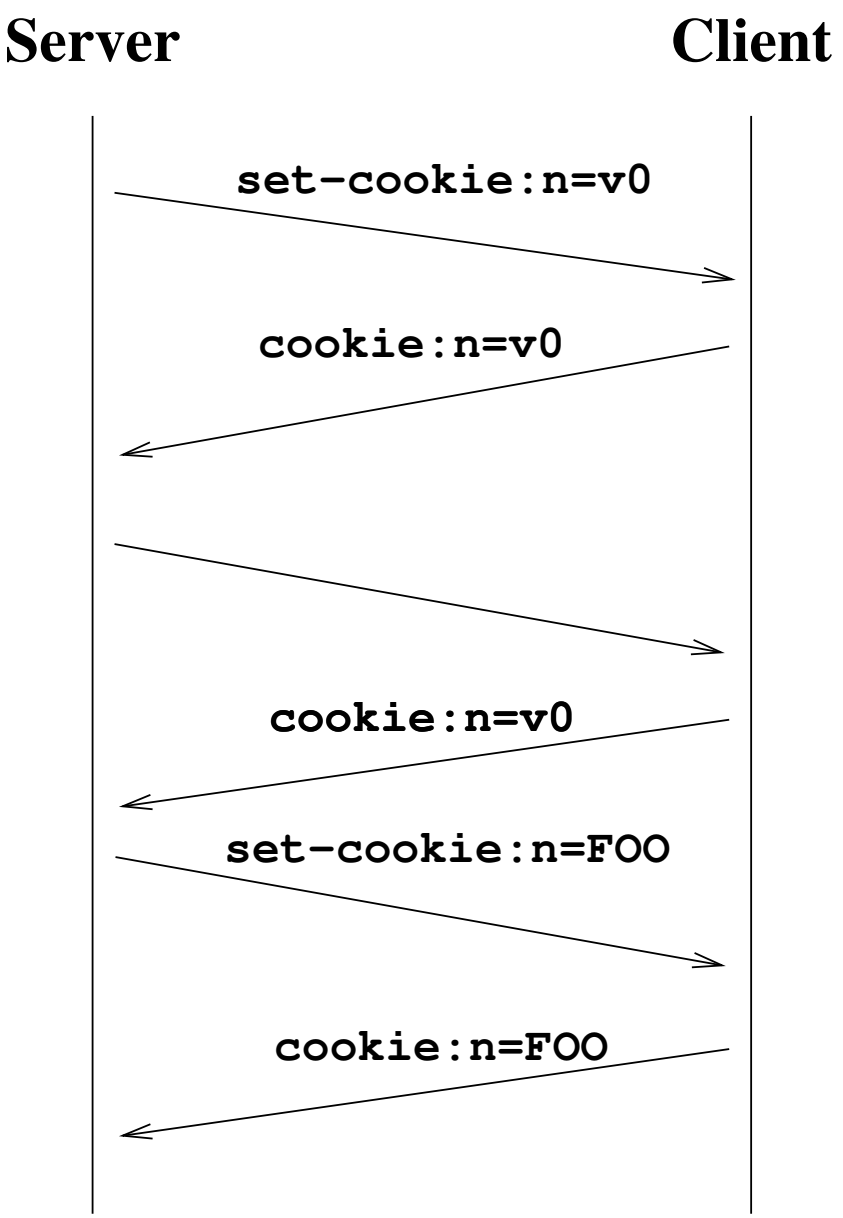
Cookies in HTTP

Siehe RFC 2109

- S: Response-Header `Set-Cookie` (mit beliebiger Antwort)
- C: Request-Header `Cookie`

Allgemeiner Ablauf

- Server sendet `Set-Cookie` um Session zu beginnen
- Client sendet `Cookie` um Session fortzuführen
- Beide können (Set-)Cookie-Header ignorieren



Set-Cookie

(+ kennzeichnet erforderliche Information)

⟨*set-cookie*⟩ ::= `Set-Cookie:`⟨*cookie-data*⟩(⟨*cookie-data*⟩)*

⟨*cookie-data*⟩ ::= ⟨*name*⟩`=`⟨*value*⟩(`;`⟨*cookie-av*⟩)* Name des Cookies

⟨*value*⟩ ::= ⟨*word*⟩ | ⟨*quoted-string*⟩

⟨*cookie-av*⟩ ::=	`Comment=`⟨*value*⟩	Dokumentation für Benutzer
	| `Domain=`⟨*value*⟩	Empfänger des Cookies (Rechner)
	| `Max-Age=`⟨*value*⟩	Gültigkeit des Cookies (Sek)
	| `Path=`⟨*value*⟩	Empfänger des Cookies (Dokumentenpfad)
	| `Secure`	
	| `Version=`⟨*number*⟩ +	Immer 1

Standardwerte

wenn `Set-Cookie` in Antwort von `http://`*`host.domain/pfad/dokument`* (wobei *`dokument`* keinen `/` enthält und *`host`* keinen `.` enthält)

`Domain=.`*`domain`*

`Max-Age` Cookie wird verworfen bei Abbruch des Browsers
(erzwungen durch negativen Wert)

`Path=`*`pfad`*

Erlaubte Werte

`Domain` Suffix von *`host.domain`*; beginnend mit Punkt; mindestens ein Punkt enthalten; kein Punkt im verbleibenden Präfix von *`host`*

`Path` Präfix von *`pfad`*

Client sollte zurückweisen, falls nicht erfüllt

Identifikation eines Cookie

- durch ⟨*name*⟩, `Domain` und `Path`;
- erneutes Senden überschreibt

Cookie

$$
\begin{array}{lcl}
\langle \textit{cookie} \rangle & ::= & \texttt{Cookie:}\langle \textit{cookie-version} \rangle \langle \textit{cookie-value} \rangle ((\texttt{;} \mid \texttt{,}) \langle \textit{cookie-value} \rangle) \\
\langle \textit{cookie-value} \rangle & ::= & \langle \textit{name} \rangle \texttt{=} \langle \textit{value} \rangle \ [\texttt{;}\langle \textit{path} \rangle] \ [\texttt{;}\langle \textit{domain} \rangle] \\
\langle \textit{cookie-version} \rangle & ::= & \texttt{\$Version=}\langle \textit{value} \rangle \\
\langle \textit{path} \rangle & ::= & \texttt{\$Path=}\langle \textit{value} \rangle \\
\langle \textit{domain} \rangle & ::= & \texttt{\$Domain=}\langle \textit{value} \rangle
\end{array}
$$

Versenden von Cookie mit Anfrage

an `http://`*host*`.`*domain*`/`*pfad*`/`*dokument*, falls

- `Domain` Attribut des Cookies ist *domain* **UND**
- `Path` Attribut des Cookies ist Präfix von *pfad* **UND**
- Cookie ist noch nicht abgelaufen

Browser versendet **alle** für die Anfrage gültigen Cookies, sortiert nach `Path`

Probleme mit Cookies

- Caching
- Sicherheit
- Grösse und Anzahl
- Zuverlässigkeit