



Netzprogrammierung Hypertext Transfer Protocol HTTP

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
<http://www.robert-tolksdorf.de>



Überblick

- HTTP
- Anfragen
- Kopfzeilen
- Antworten
- HTTPS

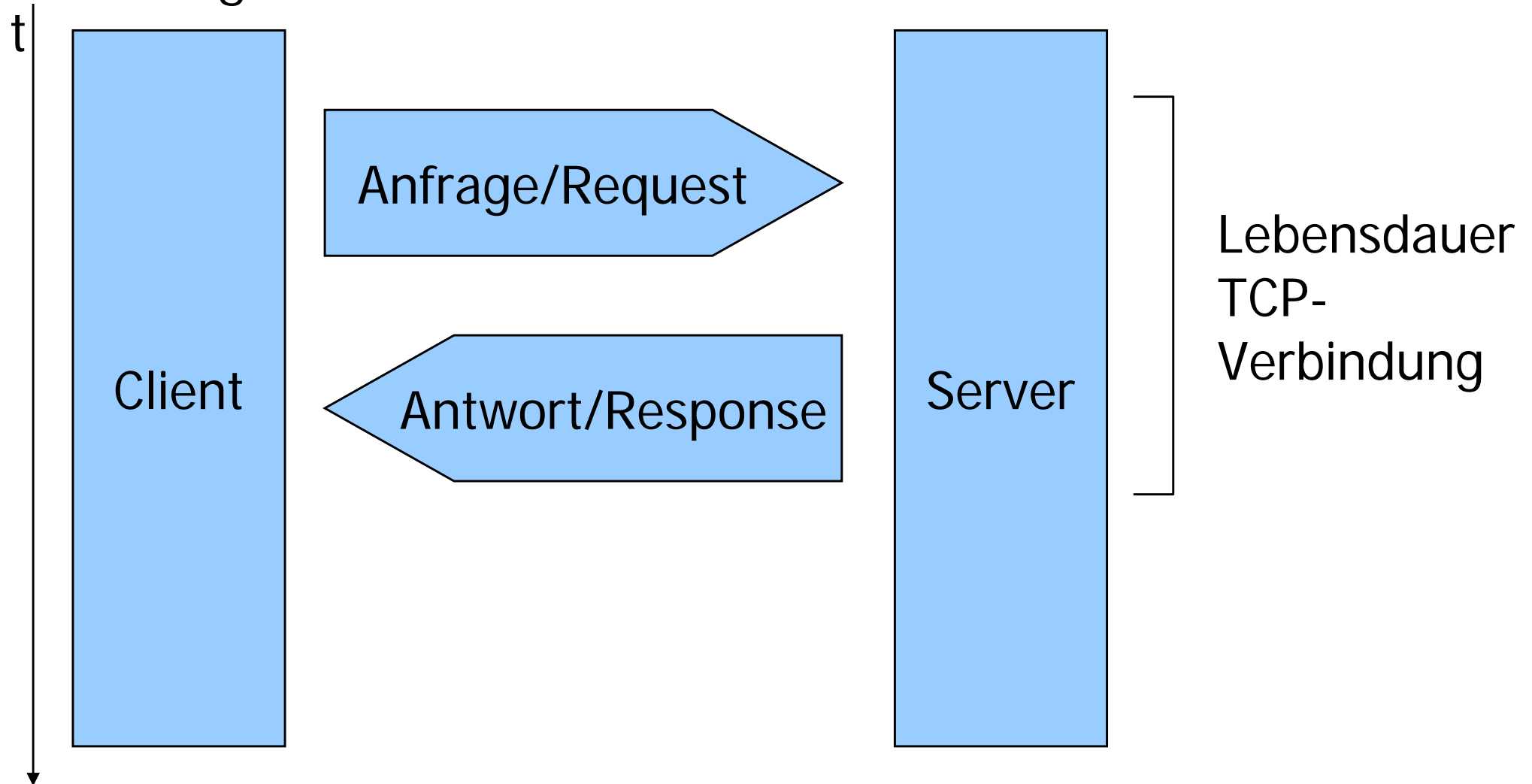


Hypertext Transfer Protocol HTTP (Überblick)

Hypertext Transfer Protocol

- Aufgabe:
Transfer von Informationen zwischen Web-Servern und Clients
- Port:
80 ist für HTTP reserviert
- Transportprotokoll:
TCP
- Protokoll:
R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee. *Hypertext Transfer Protocol - HTTP/1.1*. RFC 2616,
<http://www.ietf.org/rfc/rfc2616.txt>

- Zustandsloses Protokoll
- Anfrage mit Antwort beantwortet



Beispiel: HTTP Protokoll

Client

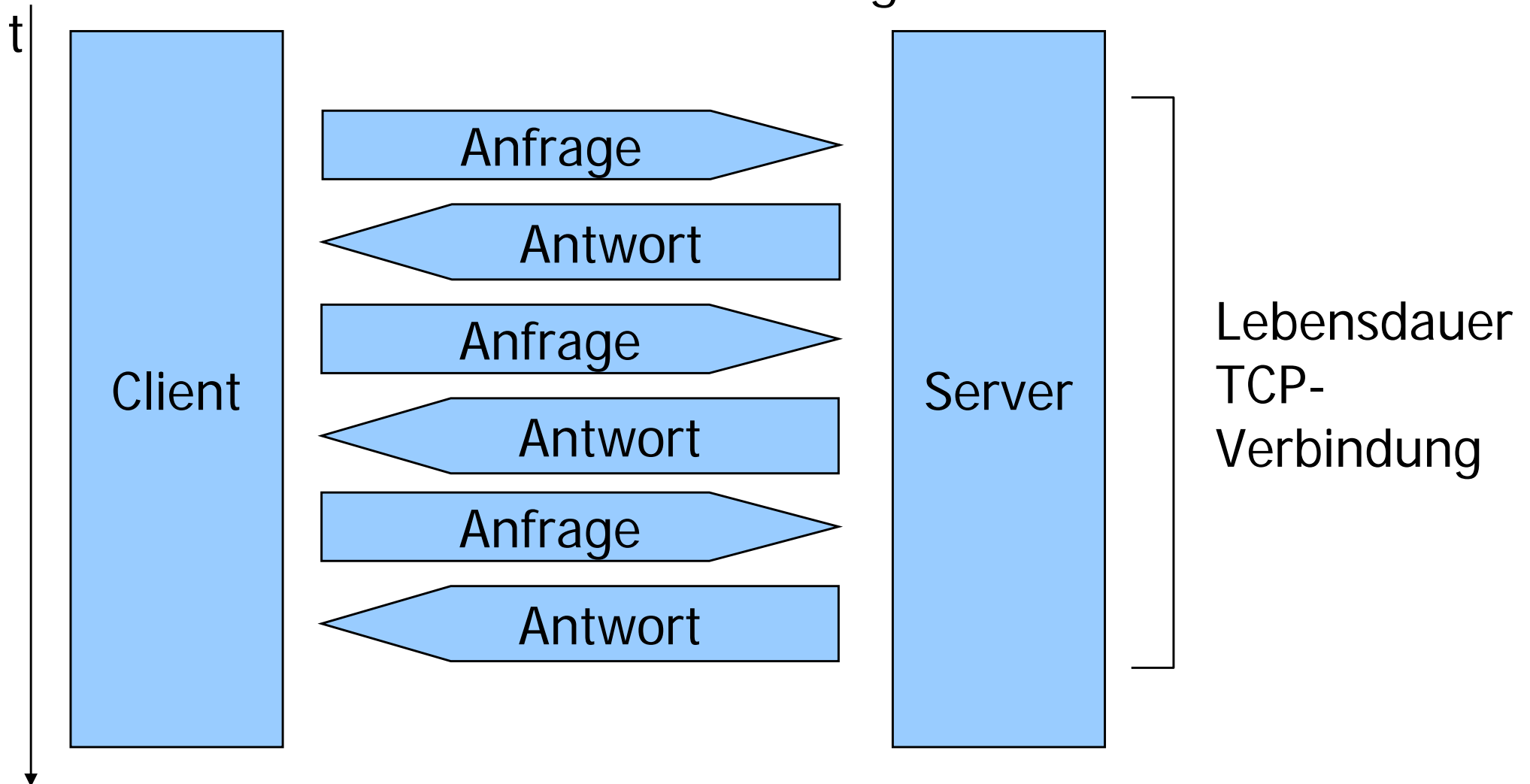
```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.04Gold (Win95; I)
Host: megababe.isdn:80
Accept: image/gif, image/jpeg, image/pjpeg, */*
```

Server

```
HTTP/1.0 200 OK
Last-Modified: Sun, 15 Mar 1998 11:26:50 GMT
MIME-Version: 1.0
Date: Fri, 20 Mar 1998 16:43:11 GMT
Server: Roxen-Challenger/1.2beta1
Content-type: text/html
Content-length: 2990

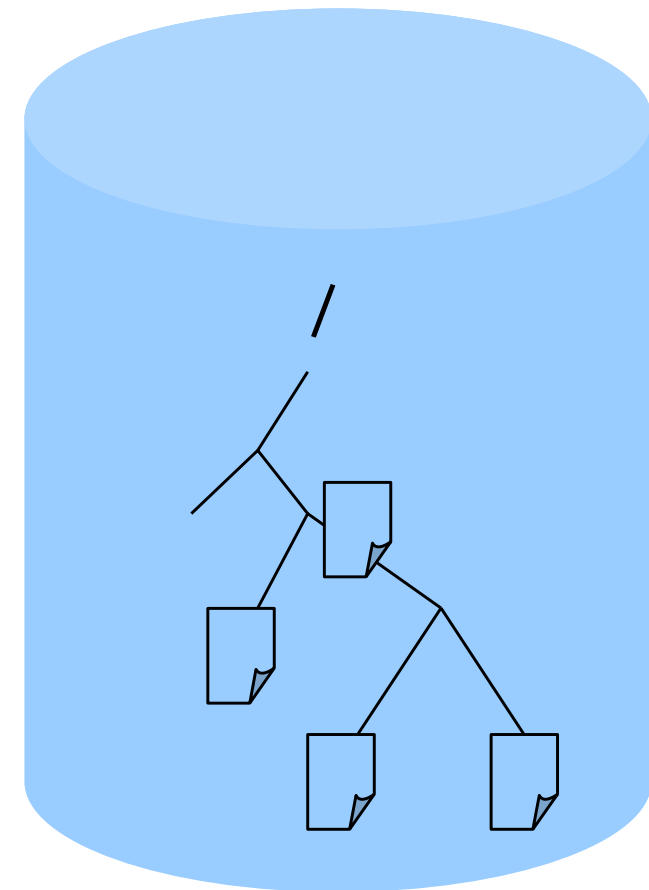
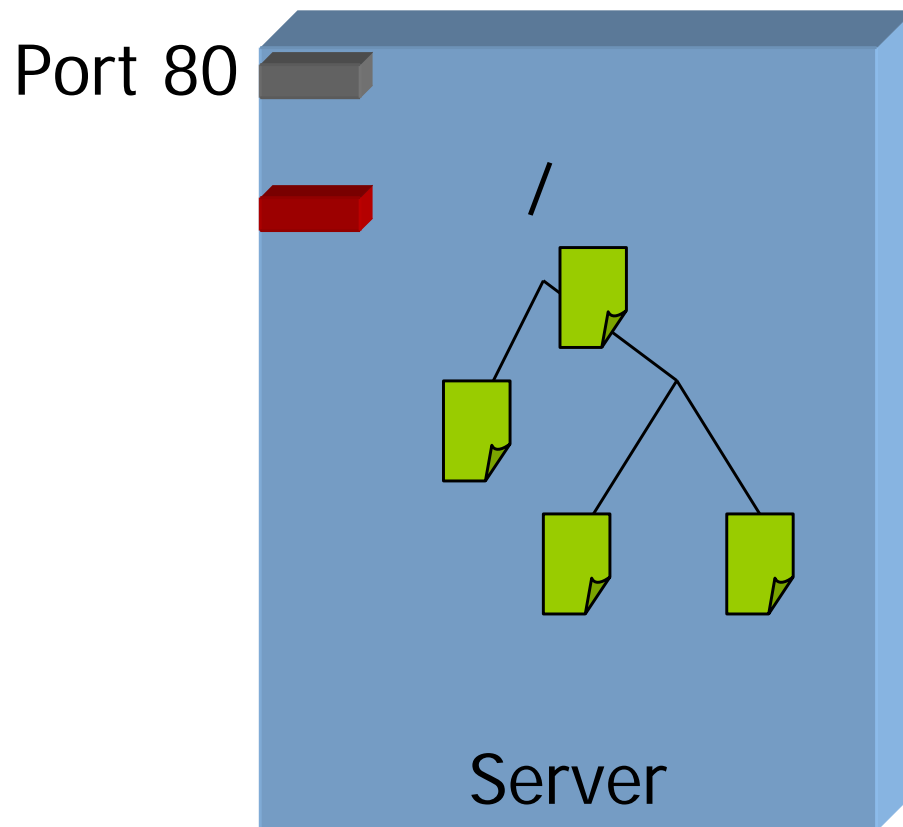
<HTML><HEAD><TITLE>TU Berlin ---
```

- HTTP 1.1 erweitert Protokoll um interaktionslange Lebensdauer der TCP Verbindung



Aufbau Web-Server

- Web-Server wartet auf Verbindungen
- Beantwortet Nachfragen nach Ressourcen bzgl. des Web-Server Verzeichnisbaums mit Dateien des verwendeten Dateisystembaums





Anfragen

Aufbau Anfrage

- Anfrage besteht aus
 - Anfragemethode
 - Anfragebeschreibung durch Kopfzeilen
 - Allgemeine Beschreibungen
 - Anfragespezifische Beschreibungen
 - Beschreibung eventuell beiliegenden Inhalts
 - Leerzeile
 - Eventueller Inhalt
- Beispiel:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.04Gold (Win95; I)
Host: megababe.isdn:80
Accept: image/gif, image/jpeg, image/pjpeg, */*
```

Anfragen in HTTP

- Format: *Methode Ressource* HTTP/x.y
- Resource ist
 - Absoluter Pfad im Server-Verzeichnisbaum
 - Voll-qualifizierte URL bei Anfrage an Proxy (s.u.)
 - *, Authority bei bestimmten Methoden
- GET Methode
 - Anforderung einer Informationseinheit vom Server
 - GET /Style/CSS/ HTTP/1.1 an Server www.w3.org
 - Beantwortet mit Code, Kopfzeilen, Inhalt

- Aufgabe:
Holen Sie die Eingangsseite eines Web-Servers
 - Bauen Sie dazu einen TCP Socket zu Port 80 auf
 - Schicken Sie die Zeile `GET / HTTP/1.0` zu dem Server
 - Lesen Sie alle Antwortzeilen

HttpClient/1

```
import java.io.*;
import java.net.*;

public class HttpClient {

    public static void main(String[] argv) {
        Socket socket;
        String line;

        try {
            socket = new Socket(argv[0],80);
        } catch (IOException iOExc) {
            System.err.println("Problem bei der Verbindungsaufnahme\n" +
                               iOExc.getMessage());
            return;
        }
    }
}
```

```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    // Früher: pw.println("GET / HTTP/1.0\n");
    pw.println("GET / HTTP/1.1\nHost: " + argv[0] + "\n");
    pw.flush();
    BufferedReader br = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
    while (true) {
        line = br.readLine();
        if (line == null) {
            break;
        } else { System.out.println(line); }
    }
} catch (IOException iOExc) {
    System.err.println("Problem beim Lesen\n" + iOExc.getMessage());
}
}
```

HttpClient/3

```
>java HttpClient www.inf.fu-berlin.de
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2005 11:39:39 GMT
Server: Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux)
      mod_perl/1.29 PHP/4.3.10-16
Last-Modified: Wed, 02 Nov 2005 09:38:29 GMT
ETag: "4814c-19dc-43688915"
Accept-Ranges: bytes
Content-Length: 6620
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
<html>
```

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="http://www.mi.fu-berlin.de/styles/homepage.css" >
```

```
  <title>Fachbereich Mathematik und Informatik</title>
```

Anfragen in HTTP

- HEAD Methode
 - Anforderung der Beschreibung einer Informationseinheit vom Server
 - HEAD /Style/CSS/ HTTP/1.1 an Server www.w3.org
 - HEAD <http://www.w3.org/Style/CSS/> HTTP/1.1 an Proxy http-proxy.fu-berlin.de
 - Beantwortet mit Code, Kopfzeilen

HttpHeadClient/1

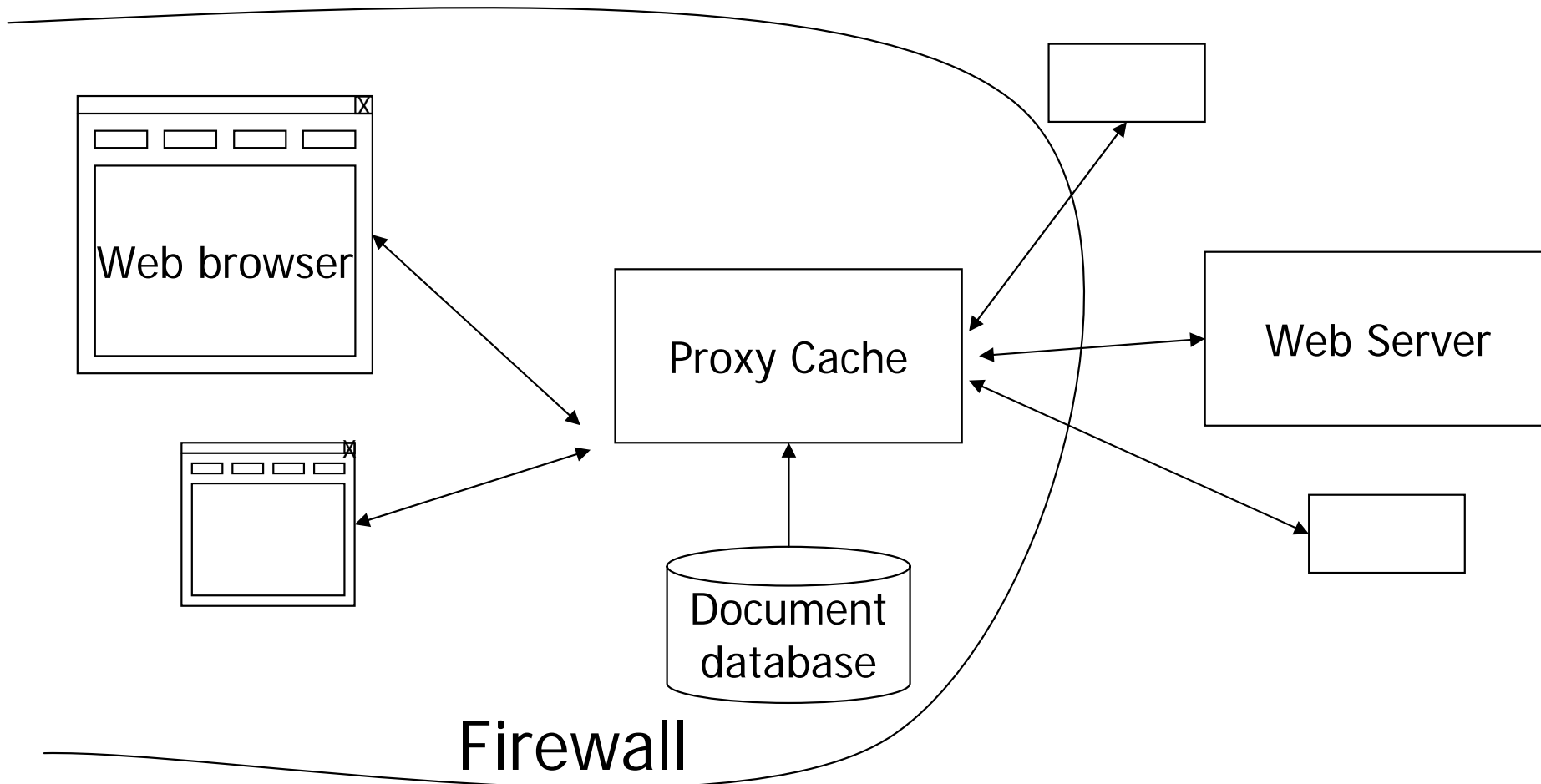
```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    pw.println("HEAD / HTTP/1.1\nHost: "+argv[0]+"\\n");
    pw.flush();
    BufferedReader br = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
    while (true) {
        line =br.readLine();
        if (line==null) {
            break;
        } else {System.out.println(line); }
    }
} catch (IOException iOExc) {
    System.err.println("Problem beim Lesen\\n"+ iOExc.getMessage());
}
}
```

HttpHeadClient/2

```
>java HttpHeadClient www.inf.fu-berlin.de
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2005 12:31:47 GMT
Server: Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux)
      mod_perl/1.29 PHP/4.3.10-16
Last-Modified: Wed, 02 Nov 2005 09:38:29 GMT
ETag: "4814c-19dc-43688915"
Accept-Ranges: bytes
Content-Length: 6620
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
>
```

Proxy-Server



- <http://www.zedat.fu-berlin.de/services/proxy.html>: „Der HTTP-Proxy Server der FU-Berlin ist unter der Adresse http-proxy.fu-berlin.de auf Port 80 zu erreichen.“

Anfragen über Proxy

- Bei Anfrage über Proxy ist Ressource komplett identifiziert
- GET Methode
 - Anforderung einer Informationseinheit vom Server
 - GET <http://www.w3.org/Style/CSS/> HTTP/1.1 an Proxy http-proxy.fu-berlin.de
 - Beantwortet mit Code, Kopfzeilen, Inhalt
- HEAD Methode
 - Anforderung der Beschreibung einer Informationseinheit vom Server
 - HEAD <http://www.w3.org/Style/CSS/> HTTP/1.1 an Proxy http-proxy.fu-berlin.de
 - Beantwortet mit Code, Kopfzeilen

HttpProxyClient/1

```
import java.io.*;
import java.net.*;

public class HttpProxyClient {

    public static void main(String[] argv) {
        Socket socket;
        String line;

        try {
            socket = new Socket("http-proxy.fu-berlin.de",80);
        } catch (IOException iOExc) {
            System.err.println("Problem bei der Verbindungsaufnahme\n" +
                iOExc.getMessage());
            return;
        }
    }
}
```

HttpProxyClient/2

```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    pw.println("GET http://" + argv[0] + argv[1] + " HTTP/1.0\n");
    pw.flush();
    BufferedReader br = new BufferedReader(
        new
        InputStreamReader(socket.getInputStream()));
    while (true) {
        line = br.readLine();
        if (line == null) {
            break;
        } else {
            System.out.println(line);
        }
    }
} catch (IOException iOExc) {
    System.err.println("Problem beim Lesen\n" + iOExc.getMessage());
}
}
```

HttpProxyClient/3

```
1>java HttpProxyClient www.spiegel.de /netzwelt
HTTP/1.0 200 OK
Date: Fri, 25 Nov 2005 12:49:36 GMT
Cache-Control: max-age=120
Expires: Fri, 25 Nov 2005 12:51:36 GMT
P3P: policyref="http://www.spiegel.de/w3c/p3p.xml", CP="NOI DSP CURa
      ADMa DEVa TAIi PSAi PSDi OUR STP IND UNI COM NAV INT STA PRE"
Content-Type: text/html
X-Cache: MISS from Inxc-581.srv.mediaways.net, MISS from
      www.spiegel.de
Via: 1.0 www.spiegel.de
X-Cache: MISS from Inxc-375.ftu.mediaways.net
X-Cache: MISS from squid.zedat.fu-berlin.de
Proxy-Connection: close
```

```
<!-- Vignette StoryServer 5.0 Fri Nov 25 13:16:55 2005 -->
<!-- 23- -->
<html>
<head>
```

HttpProxyClient/4

```
2>java HttpProxyClient www.spiegel.de /netzwelt
HTTP/1.0 200 OK
Date: Fri, 25 Nov 2005 12:52:27 GMT
Cache-Control: max-age=120
Expires: Fri, 25 Nov 2005 12:54:27 GMT
P3P: policyref="http://www.spiegel.de/w3c/p3p.xml", CP="NOI DSP CURa
      ADMa DEVa TAIi PSAi PSDi OUR STP IND UNI COM NAV INT STA PRE"
Content-Type: text/html
X-Cache: MISS from Inxc-582.srv.mediaways.net, MISS from
      www.spiegel.de
Via: 1.0 www.spiegel.de
X-Cache: MISS from Inxc-092.ftu.mediaways.net
X-Cache: HIT from squid.zedat.fu-berlin.de
Proxy-Connection: close
```

```
<!-- Vignette StoryServer 5.0 Fri Nov 25 13:16:55 2005 -->
<!-- 23- -->
<html>
<head>
```

Weitere Anfragen in HTTP

- PUT
 - Abspeichern einer Informationseinheit auf einem Server
 - PUT /index.html HTTP/1.1
 - Beantwortet mit Code, Kopfzeilen
- POST
 - Hinzufügen von Informationen zu einer Informationseinheit
 - POST /speichere.cgi HTTP/1.1
Daten daten daten
 - Beantwortet mit Code, Kopfzeilen, eventuell Inhalt
- DELETE
 - Löschen einer Informationseinheit auf einem Server
 - DELETE /index.html HTTP/1.1
 - Beantwortet mit Code, Kopfzeilen

Weitere Anfragen in HTTP

- TRACE
 - Server schickt erhaltenen Inhalt zurück
- CONNECT
 - Sagt Proxy, dass er Tunnel aufbauen soll
 - Tunnel: Verpacken eines Protokolls A in ein anderes Protokoll B, so dass die Anwendung A spricht, aber B benutzt

Weitere Anfragen in HTTP

- OPTIONS

- Informationen über Fähigkeiten des Servers
- Überträgt alle Allow-Kopfzeilen

- Anfrage:

OPTIONS * HTTP/1.1

Host: www.inf.fu-berlin.de

- Antwort:

HTTP/1.1 200 OK

Date: Tue, 25 Nov 2003 11:29:16 GMT

Server: Apache/1.3.26 Ben-SSL/1.48 (Unix) Debian

GNU/Linux mod_perl/1.26 PHP/4.1.2

Content-Length: 0

Allow: GET, HEAD, OPTIONS, TRACE



Anfrage Varianten

Anfrage Kopfzeilen

- Host: *Name*
Aus der URL ermittelter Name des Server-Rechners von dem angefordert wird. *Einziger Pflichtkopfzeile in HTTP 1.1*
- Date: Tue, 15 Nov 1994 08:12:31 GMT
Datum des Abschickens der Anfrage im RFC 1123 Format
- Connection: close
Verbindung nach Ergebnisübermittlung abbauen

- If-Modified-Since: *Datum*
Änderung der Informationseinheit seit Datum
 - Ja: 200 und Inhalt schicken
 - Nein: 304 und Inhalt nicht schicken
- If-Unmodified-Since: *Datum*
Änderung der Informationseinheit seit Datum
 - Ja: 412 und nicht verarbeiten
 - Nein: Normal verarbeiten (als sei If-Unmodified-Since: nicht vorhanden)

HttpModClient/1

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;

public class HttpModClient {

    public static void main(String[] argv) {
        Socket socket;
        String line;

        try {
            socket = new Socket(argv[0],80);
        } catch (IOException iOExc) {
            System.err.println("Problem bei der Verbindungsaufnahme\n"+
                               iOExc.getMessage());
            return;
        }
    }
}
```

HttpModClient/2

```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    pw.println("GET / HTTP/1.0");
    SimpleDateFormat rfc1123 =
        new SimpleDateFormat("EEE, dd MMM yyyy hh:mm:ss z",
                             Locale.US);
    Calendar cal = Calendar.getInstance();
    cal.setTime(new Date());
    cal.add(Calendar.DATE, Integer.parseInt(argv[1]));
    pw.println("If-Modified-Since: " +
               rfc1123.format(cal.getTime()) + "\n");
    pw.flush();
    BufferedReader br = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));
    while (true) {
```

...

HttpModClient/3

```
>java HttpModClient www.inf.fu-berlin.de -20
HTTP/1.1 304 Not Modified
Date: Fri, 25 Nov 2005 14:53:30 GMT
Server: Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux) mod_perl/1.29 PHP/4.3.10-16
Connection: close
ETag: "4814c-19dc-43688915"
```

```
>java HttpModClient www.inf.fu-berlin.de -30
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2005 14:53:46 GMT
Server: Apache/1.3.33 Ben-SSL/1.55 (Debian GNU/Linux) mod_perl/1.29 PHP/4.3.10-16
Last-Modified: Wed, 02 Nov 2005 09:38:29 GMT
ETag: "4814c-19dc-43688915"
Accept-Ranges: bytes
Content-Length: 6620
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
<html>
<head>
```

Anfrage Kopfzeilen

- Range: *Bytebereich*

Nur Teile der Information anfordern, Antwort ist dann 216

Range: bytes=500-999

- pw.println("GET / HTTP/1.1\nHost: " + argv[0]);
pw.println("Range: bytes=" + argv[1] + "-" + argv[2] + "\n");
- java HttpClientRange www.fu-berlin.de 1 100
- HTTP/1.1 206 Partial Content
Date: Mon, 01 Oct 2007 15:13:35 GMT
Server: Apache/2.0.59 (Linux/SuSE)
Last-Modified: Mon, 01 Oct 2007 12:34:51 GMT
ETag: "1c988fa-4148-a9acf0c0,"
Accept-Ranges: bytes
Content-Length: 100
Content-Range: bytes 1-100/16712
Content-Type: text/html; charset=utf-8

Anfrage Kopfzeilen

- From: *Mailadresse*
Nutzer – real nie genutzt
- User-Agent: *Produkt/Version*
Browser z.B.
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
 - Browser „maskieren“ sich
 - Zuverlässigkeit des Headers nicht gegeben
 - Man kann je nach Browser unterschiedlichen Inhalt ausgeben



Vodafone/1.0/HTC_wizard/
2.21.3.102/Mozilla/4.0
(compatible; MSIE 4.01;
Windows CE; PPC; 240x320)

- Referer: *URL*
Seite auf der ein Link auf die angeforderte Seite stand

Anfrage Kopfzeilen

- Authorization: *Nachweis*
Autorisierungsnachweis falls mit 401 angefordert
- Authorization: username="Mufasa",
 realm="testrealm@host.com",

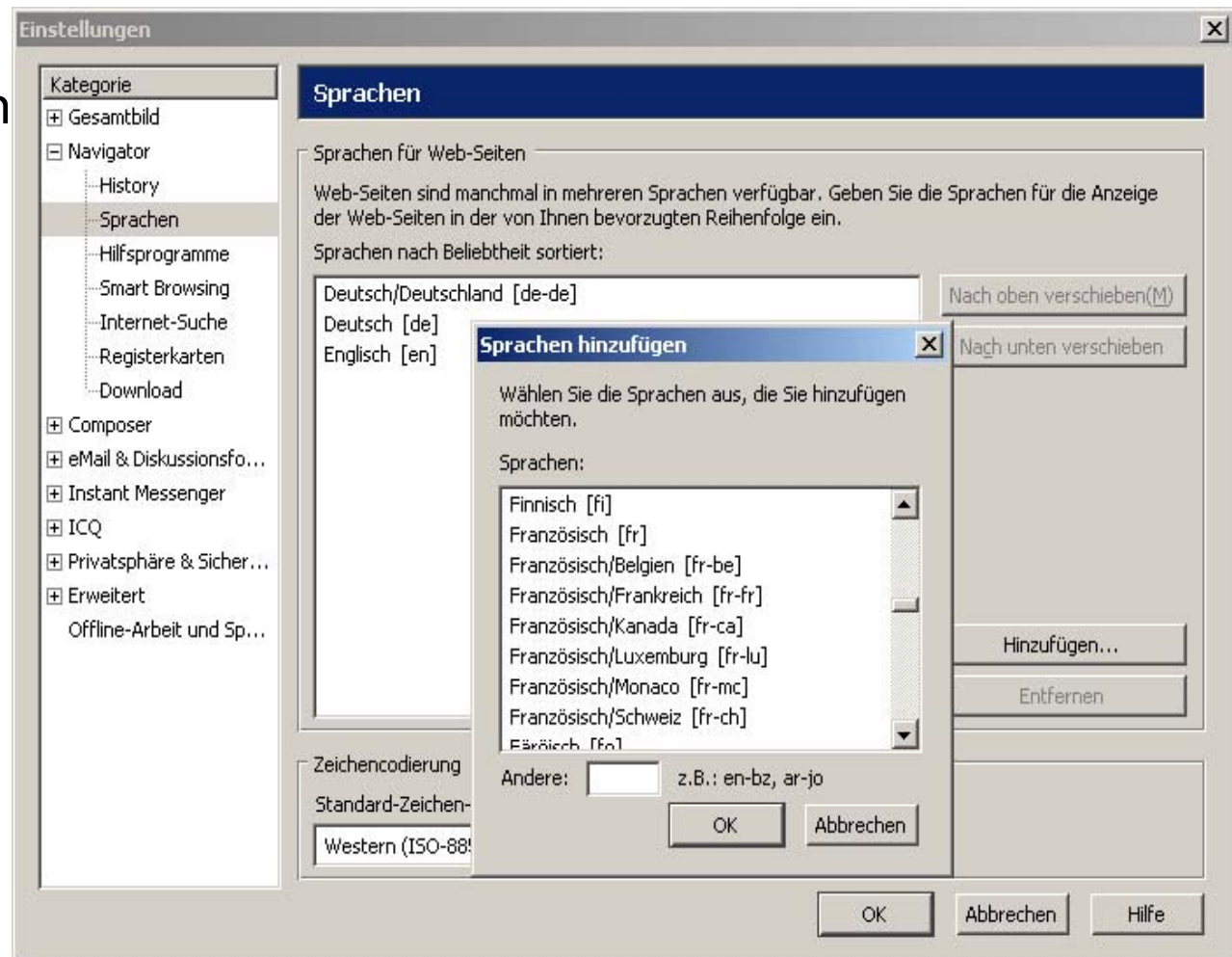
 response="6629fae49393a05397450978507c4ef1"
- Proxy-Authorization: *Nachweis*
Autorisierungsnachweis für Proxy, falls mit 407 angefordert
- Details in nächster VL
- Max-Forwards: *Anzahl*
Wie oft ein OPTIONS oder TRACE weitergeleitet werden darf
- Expect: *Token*
Client erwartet bestimmte Eigenschaften von Server/Proxy
(Falls nicht: 417)



Anfrage Anforderungen

Angeforderte Sprache

- Browser kann Präferenzen im HTTP-Request mitteilen:
GET / HTTP/1.1
Accept-Language: de, en
Oder
Accept-Language: en
- q gibt
Priorität an,
* ist
Platzhalter
- Vom Browser
abhängig:
- Weiterhin
anforderbar:
Zeichen-
eigenschaften



Angeforderte Sprache

- Browser kann Präferenzen im HTTP-Request mitteilen:
GET / HTTP/1.1
Accept-Language: de, en
Oder
Accept-Language: en-us; q=0.75, en; q=0.5; *; q=0.25
- q gibt
Priorität an,
* ist
Platzhalter
- Vom Browser
abhängig:
- Weiterhin
anforderbar:
Zeichen-
eigenschaften

- Accept-Charset: Zeichensatz

ISO-8859-1	ISO-8859-2	ISO-8859-3
ISO-8859-4	ISO-8859-5	ISO-8859-6
ISO-8859-7	ISO-8859-8	ISO-8859-9
ISO-2022-JP	ISO-2022-JP-2	ISO-2022-KR
UNICODE-1-1	UNICODE-1-1-UTF-7	UNICODE-1-1-UTF-8
US-ASCII	...	

- Zeichenrepertoire (Character Set, Abstract Character Repertoire, ACS)
 - Eine Menge von Zeichen
 - Definiert durch Namen und Beispiele
 - {Pfund (£), Zett (Z), Ypsilon (Y), Herz (♥)}
 - Keine Ordnung, keine Codierung
- Zeichencode (Coded Character Set, CCS)
 - Abbildung(en) Zeichen → Zeichenposition
 - Z → 5A, ç → FEA5 (Khah)
 - z.B. UNICODE, ISO 8859-1

UNICODE: Braille

		288	289	28A	28B	28C	28D	28E	28F										
	28B3	28C3										280	281	282	283	284	285	286	287
	⠠⠠	⠠⠠									0	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									1	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									2	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	28B4	28C4									3	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									4	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									5	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	28B5	28C5									6	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									7	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									8	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	28B6	28C6									9	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									A	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									B	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	28B7	28C7									C	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									D	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠									E	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	28B8	28C8									F	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠
	⠠⠠	⠠⠠										⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠	⠠⠠

0F36	Tibetan	0F7E
0F36 ✧ TIBETAN MARK CARET -DZUD RTAGS BZHI MIG CAN • marks point of text insertion or annotation	0F5C ལྷོ TIBETAN LETTER DZHA ≡ 0F5B ལྷོ 0FB7 ལྷོ	
0F37 ◦ TIBETAN MARK NGAS BZUNG SGOR RTAGS • emphasis; used like underlining	0F5D ཨ TIBETAN LETTER WA	
0F38 ར TIBETAN MARK CHE MGO	0F5E ལྷོ TIBETAN LETTER ZHA	
0F39 འ TIBETAN MARK TSA -PHRU • a lenition mark	0F5F ལྷོ TIBETAN LETTER ZA	
	0F60 ལྷོ TIBETAN LETTER -A	
	0F61 ལྷོ TIBETAN LETTER YA	
	0F62 ལྷོ TIBETAN LETTER RA • when followed by a subjoined letter = ra mgo	
Paired punctuation	0F63 ལྷོ TIBETAN LETTER LA	
0F3A ལྷོ TIBETAN MARK GUG RTAGS GYON	0F64 ལྷོ TIBETAN LETTER SHA	
0F3B ལྷོ TIBETAN MARK GUG RTAGS GYAS • brackets	0F65 ལྷོ TIBETAN LETTER SSA = reversed sha	
0F3C } TIBETAN MARK ANG KHANG GYON	0F66 ལྷོ TIBETAN LETTER SA	
0F3D { TIBETAN MARK ANG KHANG GYAS • used for bracketing with a roof over	0F67 ལྷོ TIBETAN LETTER HA	
	0F68 ལྷོ TIBETAN LETTER A • base for dependent vowels	
Astrological signs	0F69 ལྷོ TIBETAN LETTER KSSA ≡ 0F40 ལྷོ 0FB5 ལྷོ	
0F3E ལྷོ TIBETAN SIGN YAR TSHES	0F6A ལྷོ TIBETAN LETTER FIXED-FORM RA • used only in transliteration and transcription	
0F3F ལྷོ TIBETAN SIGN MAR TSHES • marks which combine with digits		
Consonants	Dependent vowel signs	
0F40 ལྷོ TIBETAN LETTER KA	0F71 ལྷོ TIBETAN VOWEL SIGN AA = a-chung • common, vowel-lengthening mark	
0F41 ལྷོ TIBETAN LETTER KHA	0F72 ལྷོ TIBETAN VOWEL SIGN I	
0F42 ལྷོ TIBETAN LETTER GA	0F73 ལྷོ TIBETAN VOWEL SIGN II • use of this character is discouraged	
0F43 ལྷོ TIBETAN LETTER GHA ≡ 0F42 ལྷོ 0FB7 ལྷོ		
0F44 ལྷོ TIBETAN LETTER NGA		
0F45 ལྷོ TIBETAN LETTER CA		

Zeicheneigenschaften

- Zeichenkodierung (Encoding)
 - Character Encoding Form (CEF)
 - Abbildung einer Zeichenfolge auf Strom gleichgroßer Codes
 - z.B.

005A	FEA5
------	------
 - Character Encoding Scheme (CES)
 - Abbildung einer Zeichenfolge auf einen Bytestrom
 - z.B.

5A	00	A5	FE
----	----	----	----
- Zeichensatz
 - Bedeutung unklar, kann Repertoire, Code oder Kodierung meinen
- „charset“ bei Accept-charset
 - meint Encoding!

Anforderung mit Prioritäten

- Browser kann Präferenzen im HTTP-Request mitteilen:
GET / HTTP/1.1
Accept-Charset: i so-8859-1, utf-8; q=0.75, *; q=0.5
- q gibt Priorität an, * ist Platzhalter
- Vom Browser abhängig:
 - Microsoft IE: Keine Angabe
 - Netscape 4.72: i so-8859-1, *, utf-8
 - NS 6.2: I S0-8859-1, utf-8; q=0.66, *; q=0.66
 - Opera 6.0:
wi ndows-1252; q=1.0, utf-8; q=1.0, utf-16; q=1.0,
i so-8859-1; q=0.6, *; q=0.1

Angeforderte Kodierung

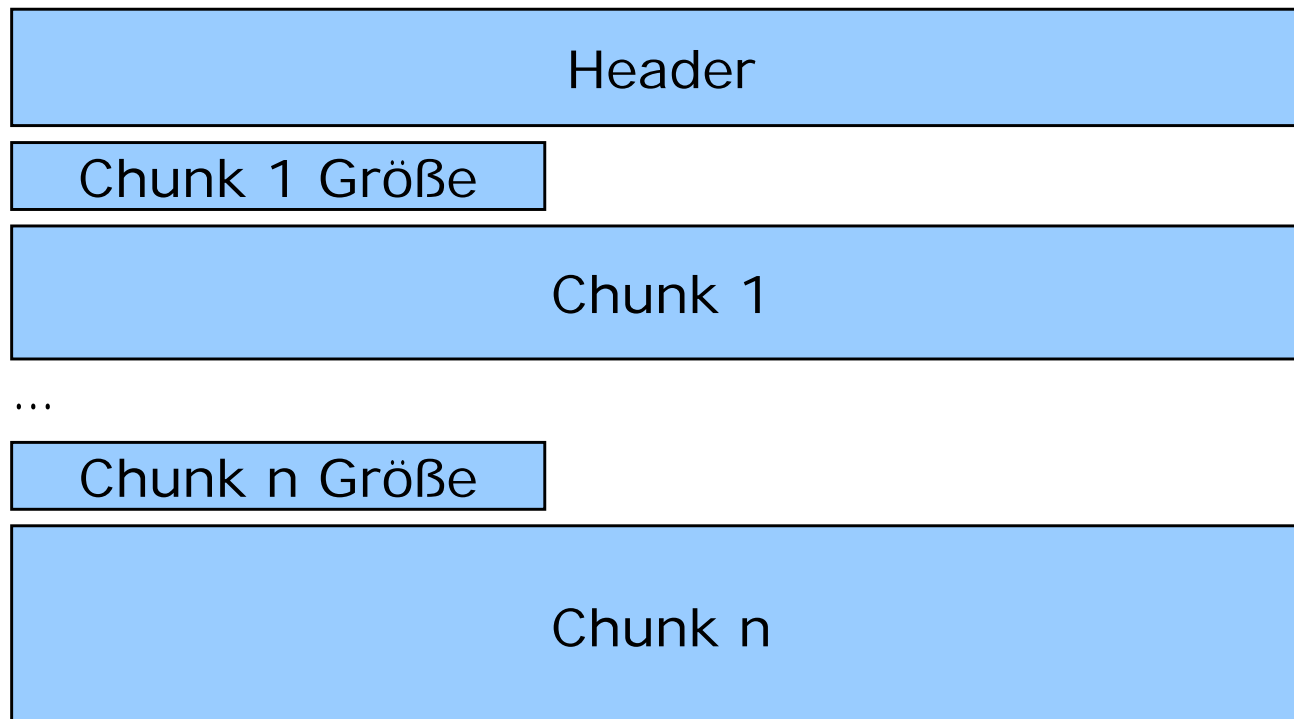
- Accept-Encoding: *Kodierung*
 - identity: Mitteilung unkodiert geschickt
 - gzip, compress, deflate: Komprimierte Übertragung
 - gzip: Lempel-Ziv LZ77 mit 32 Bit Prüfsumme, RFC 1952. gzip Programm
 - compress: adaptives Lempel-Ziv-Welch (LZW). compress Programm
 - deflate: "zlib"-Format, RFC 1950 zusammen mit "deflate" Kompression RFC 1951
- Man kann das Web komprimiert ausliefern

Transferencoding in HTTP

- Zusätzliche Transferencoding verändert den Inhalt einer übermittelten Information
- Beispiel: Komprimierung durch gzip-Verfahren
- In der Anfrage
GET / HTTP/1.1
Accept-Encoding: compress; q=0.5, gzip; q=1.0
- In der Antwort
200 OK HTTP/1.1
Content-Encoding: gzip
oder
200 OK HTTP/1.1
Transfer-Encoding: gzip
- Kann auf Transportweg (Proxies) geändert werden

Transfer-Encoding chunked

- chunked:
 - Wenn die Anzahl der zu übertragenden Zeichen nicht vorher feststeht kann Content-Length nicht gesetzt werden
 - Client müsste auf Timeout auf dem Socket bei Pausen warten bevor er schon übertragene Inhalte verarbeitet
 - Chunked: Inhalt wird in Häppchen übertragen:



Transfer-Encoding chunked

HTTP/1.1 200 OK

Date: Fri, 28 Sep 2007 15:05:01 GMT

Server: Apache

Content-Type: text/html; charset=utf-8

Transfer-Encoding: chunked

3857

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >
```

```
<html> <head>
```

```
<title>FOCUS Online Nachrichten - News aus den Ressorts Politik, Finanzen,  
Gesun
```

```
dheit, Kultur, Wissen, Jobs, Digital, Reise, Sport und Auto</title>
```

```
<meta http-equiv="content-language" content="de"/> <meta  
name="keywords" content="FOCUS Online, FOCUS, FOCUS MAGAZIN,  
www.focus.de, Nachrichten, aktuell, aktuel
```

...

```
</div> </div>
```

4000

```
<div class="main1"> <div class="main2"> <script type="text/javascript">
```

...

Angeforderte Medienart

- Accept: *Medienart/Variante*; q=*Qualität*;
mxb=*Maximale Größe*
 - Accept: text/postscript; mxb=200000

Exkurs: Inhaltstypen

- Per HTTP können beliebige Inhalte transportiert werden, nicht nur HTML
- Multipurpose Internet Mail Extensions MIME (RFC 2045, RFC 2046) definiert ein Schema zur eindeutigen Benennung durch einen Inhaltstypen
- In HTTP in Kopfzeile Content-Type
- Format: *Typ/Untertyp*
 - text/html
 - image/jpeg
 - vnd.motorola.video

```
HTTP/1.0 200 OK
Last-Modified: Sun, 15 Mar 1998 11:26:50 GMT
MIME-Version: 1.0
Date: Fri, 20 Mar 1998 16:43:11 GMT
Server: Roxen-Challenger/1.2beta1
Content-type: text/html
Content-length: 2990

<HTML><HEAD><TITLE>TU Berlin ---
```

MIME Typen

- Acht Typen:
 - text: Text
 - text/plain, text/html, text/rtf, text/vnd.latex-z
 - image: Grafiken
 - image/png, vnd.microsoft.icon
 - video: Bewegtbilder
 - video/mpeg, video/quicktime, video/vnd.vivo
 - audio: Audiodaten
 - audio/G726-16 , audio/vnd.nokia.mobile-xmf
 - application: binäre und/oder anwendungsspezifische Daten
 - application/EDIFACT, application/vnd.ms-powerpoint
 - multipart: mehrteilige Daten
 - multipart/mixed
 - message: Nachrichten
 - message/rfc822
 - model: Daten
 - model/vrml

MIME Typen

- MIME-Typen werden von der Internet Corporation for Assigned Names and Numbers IANA verwaltet
- <http://www.iana.org/assignments/media-types/>
- Verarbeiten eines bestimmten Medientyps nach Erhalt:
 - Teil der Anwendung (siehe auch: `javax.mail.internet.MimeMessage`)
 - eventuell Unterstützung durch Betriebssystem
- Ermittlung des MIME-Typs für eine Datei:
 - Ableitung aus Endung (`javax.activation.MimetypesFileTypeMap`)
 - Ableitung aus Inhalt der Datei

Content Negotiation

- Auswahl passender Information bezüglich der Dimensionen
 - Medienart (Accept: text/html, text/plain)
 - Sprache (AcceptLanguage: en-us; q=0.75, en; q=0.5; *; q=0.25)
 - Encoding (Accept-Encoding: compress; q=0.5, gzip; q=1.0)
 - Charset (AcceptCharset: iso-8859-1, utf-8; q=0.75, *; q=0.5)
 - Angegebene Qualitätsmaße
- Server-abhängige Implementierungen
 - z.B. Schema über Dateinamen:
 - foo.en.html
 - foo.html.en
 - foo.en.html.gz

Inhalts-Kopfzeilen

- Content-Encoding: *Kodierung*
Kodierung des Inhalts
 - deflate, gzip, ...
- Content-Type: *Medienart*
Medientyp des Inhalts
 - text/html, image/gif, ..
- Content-Language: *Sprachkürzel*
Sprache des Inhalts
 - de, en, en-US
- Content-Length: *Länge*
Länge des Inhalts in Byte
- Content-Range: *Range*
Beschreibung des Ausschnitts bei Teilanforderung

Inhalts-Kopfzeilen

- Content-Location: *URI*
Inhalt ist in Antwort, der Inhalt steht aber auch an einer anderen URI
- Content-MD5: *MD5Checksum*
Message Digest für Inhalt zur Integritätsprüfung
- Expires: *Datum*
Kann nach *Datum* aus Caches gelöscht werden
- Last-Modified: *Datum*
Letzte Änderung

- Cache-Control: *Direktive*
Steuert das Caching von Anfragen und Antworten
 - no-cache: Antwort darf nicht zur Beantwortung anderer Anfragen genutzt werden
 - no-store: Antwort- oder Anfragemitteilungen dürfen nicht gespeichert werden
 - weitere: max-age, max-stale, min-fresh, no-transform, only-if-cached, public, private, must-revalidate, proxy-revalidate, s-maxage
- Pragma: no-cache
Entspricht Cache-Control: no-cache

- *Via: Protokollversion Host ...*
Weg der Nachricht, z.B. *Via: 1.0 fred, 1.1 nowhere.com*
(Apache/1.1)
- *Upgrade: Protokoll*
Wunsch nach Verwendung eines neueren Protokolls
z.B.: *Upgrade: HTTP/2.0*
- *Trailer: Trailer-Header*
Nach dem Inhalt folgen weitere Kopfzeilen geschickt
- *Warning: Freitext*
Zusätzlicher Hinweis



Antworten

Aufbau Antwort

- Antwort besteht aus
 - Antwortcode
 - Antwortbeschreibung durch Kopfzeilen
 - Allgemeine Beschreibungen
 - Antwortspezifische Beschreibungen
 - Beschreibung eventuell beiliegenden Inhalts
 - Leerzeile
 - Eventueller Inhalt
- Beispiel:

```
HTTP/1.0 200 OK
Last-Modified: Sun, 15 Mar 1998 11:26:50 GMT
MIME-Version: 1.0
Date: Fri, 20 Mar 1998 16:43:11 GMT
Server: Roxen-Challenger/1.2beta1
Content-type: text/html
Content-length: 2990

<HTML><HEAD><TITLE>TU Berlin ---
```

Antwort Codes

- 200-er Codes: Erfolgreiche Ausführung
 - **200 – OK**
GET, HEAD, POST, TRACE erfolgreich, Antwort anbei
 - 201 – Created
Erfolgreiches PUT oder POST
 - 202 – Accepted
Für spätere Ausführung vermerkt
 - 203 – Non-Authoritative Information
Metainformationen in Kopfzeilen stammen von Dritten
 - 204 – No Content
Anfrage verarbeitet, kein Antwortinhalt notwendig
 - 205 – Reset Content
Anfrage verarbeitet, Ansicht erneuern
 - **206 – Partial Content**
GET mit Teilanforderung erfolgreich, Teilantwort anbei

Antwort Codes

- 300-er Codes: Weitere Aktion des Client zur erfolgreichen Ausführung notwendig
 - 300 - Multiple Choices
Verschiedene Versionen erhältlich, Accept-Kopfzeile nicht eindeutig
 - 301 - Moved Permanently
Verschoben (Location und URI Kopfzeilen geben Auskunft)
 - 302 - Found Moved Temporarily
Verschoben (Location und URI Kopfzeilen geben Auskunft)
 - 303 See Other
Andere Resource laden (Location und URI Kopfzeilen geben Auskunft)
 - 304 - Not Modified
Bei GET mit If-Modified-Since Kopfzeile
 - 305 Use Proxy
Muss durch Proxy angesprochen werden (Adresse in Location)
 - 307 Temporary Redirect
Umleitung bei GET, HEAD

Antwort Codes

- 400-er Codes: Nicht erfolgreich, Fehler bei Client
 - 400 - Bad Request
Falsche Anfragesyntax
 - 401 - Unauthorized
Passwort notwendig
 - 403 – Forbidden
Ohne Angabe von Gründen verweigert
 - 404 - Not Found
Nicht auffindbar
 - 405 - Method Not Allowed
Methode für die Resource nicht zugelassen
 - 406 - Not Acceptable
Information vorhanden aber nicht passend zu Accept-Kopfzeilen
 - 407 Proxy Authentication Required
Zuerst Authentifizierung bei Proxy nötig, der Proxy-Authenticate Kopfzeilen mit schicken muss
 - 408 - Request Timeout
Timeout bei Übermittlung der Anfrage

Antwort Codes

- 409 Conflict
Methode steht in Konflikt mit Zustand des Servers, Client kann Konflikt aufheben
- 410 Gone
Permanent und absichtlich nicht auffindbar
- 411 Length Required
Content- Length Kopfzeile ist notwendig
- 412 Precondition Failed
Bedingungen der Anfrage (in Kopfzeilen) unerfüllbar
- 413 Request Entity Too Large
Anfrage zu groß
- 414 Request-URI Too Long
URI zu lang
- 415 Unsupported Media Type
Unbekanntes Inhaltsformat
- 416 Requested Range Not Satisfiable
Teilanforderung falsch beschrieben
- 417 Expectation Failed
Expect Kopfzeile unerfüllbar

Antwort Codes

- 500-er Codes: Nicht erfolgreich, Fehler bei Server
 - 500 - Internal Server Error
 - 501 - Not Implemented
Angeforderte Methode nicht unterstützt
 - 502 - Bad Gateway
Weiterer benutzer Server nicht erreichbar
 - 503 - Service Unavailable
Server kann Dienst gerade nicht erbringen (Retry-After
Kopfzeile)
 - 504 - Gateway Timeout
Weiterer benutzter Server antwortet nicht rechtzeitig
 - 505 HTTP Version Not Supported
Unbekannte HTTP Version

Antwort Kopfzeilen

- Server: *Produkt*
Server-Produkt
Server: CERNb-HTTPD/3.0 libwww/2.17
- Accept-Ranges: *Token*
Inwiefern der Server Teilübertragungen unterstützt
Accept-Ranges: bytes
Accept-Ranges: none
- Retry-After: *Datum*
Bei 503: Zeitpunkt zur Wiederholung der Anfrage
Retry-After: Fri, 31 Dec 1999 23:59:59 GMT
Retry-After: 120
- Age: *Sekunden*
Geschätztes Alter der Resource

Antwort Kopfzeilen

- Location: *URI*
Adresse unter der Resource aufzufinden ist
Bei 201: Adresse der neu geschaffenen Resource
Bei 3xx: URI für Umlenkung
- WWW-Authenticate: *Aufgabe*
Bei 401: Client muss sich gegenüber Server ausweisen
- Proxy-Authenticate: *Aufgabe*
Bei 407: Client muss sich gegenüber Proxy ausweisen

Web Server

- Aufgabe: Schreiben Sie einen Webserver, der alle Anfrage mit derselben HTML-Seite beantwortet.
 - Ein Webserver wartet an Port 80 auf Verbindungen
 - Er erhält über die Verbindung eine Zeile der Art *GET /Pfad*
 - Er antwortet mit HTML Code und schließt die Verbindung
 - Vor der HTML Seite muß
HTTP/1.0 200 Ok
Content-Type: text/html
Leerzeile
stehen damit der Browser sie richtig anzeigt

WebServer/1

```
import java.net.*;
import java.io.*;
public class WebServer {
    public static void main(String[] argv) {
        // Nummer des Ports von Kommandozeile (Default 80)
        int port=80;
        if (argv.length==1) {
            try {
                port=java.lang.Integer.parseInt(argv[0]);
            } catch (Exception e) {}
        }
        // Hauptprogramm
        try {
            // Server initialisieren
            ServerSocket serverSocket= new ServerSocket(port);
        }
    }
}
```

WebServer/2

```
while (true) {
    Socket connection=serverSocket.accept();
    BufferedReader br = new BufferedReader(new InputStreamReader
        (connection.getInputStream())); // Eine Zeile lesen
    String httpLine=br.readLine();
    // Antwort senden
    PrintWriter pw = new PrintWriter(connection.getOutputStream());
    pw.println("HTTP/1.0 200 Ok\n" + "Content-type: text/html\n\n" +
        "<HTML><HEAD><TITLE>Hello</TITLE></HEAD>\n" +
        "<BODY><H1>Willkommen</H1>\n" +
        "<P>Das HTTP Kommando war:\n" +
        "<PRE>\n" + httpLine + "\n</PRE>\n</BODY></HTML>\n");
    pw.flush();
    connection.close();
} // Interaktion fertig
} catch (Exception e) { System.err.println(e.getMessage()); }
}
```



Sichere HTTP Verbindungen über SSL

http vs. https...



*** ATTENTION ***



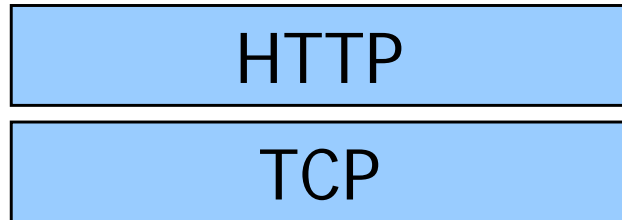
Please go to <https://10.0.0.1/>
or
to <https://luftbruecke.inf.fu-berlin.de/>

Web-Zugriff nur per SSL (Port 443).
Bitte <https://luftbruecke.inf.fu-berlin.de> benutzen.

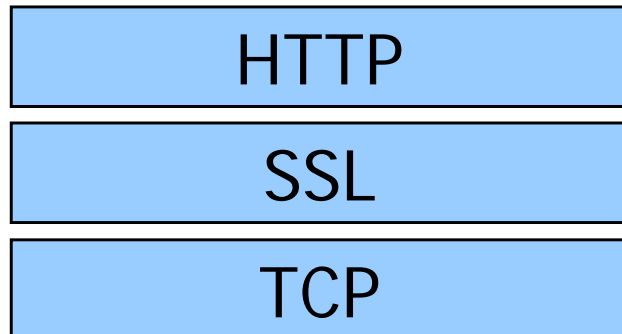


HTTP über SSL Sockets

- HTTP benutzt TCP Sockets zur Kommunikation



- Secure Sockets Layer SSL erweitert Sockets um Sicherheitsmerkmale
- HTTPS bezeichnet eine HTTP Kommunikation über solche sicheren Sockets



- Port 443 als Default-Port festgelegt

Secure Sockets Layer SSL

- 1994 von Netscape entwickelt
- Basierend auf SSL 3.0:
IETF-Standard
Transport Layer Security (TLS)
- T. Dierks, C. Allen. RFC 2246 - The TLS Protocol Version 1.0 <http://www.ietf.org/rfc/rfc2246.txt>
- Nicht auf HTTP beschränkt, auch andere Anwendungsprotokolle können darüber abgewickelt werden

Secure Sockets Layer SSL

- SSL
 - Sichert die Integrität von übertragenen Daten durch Verhinderung durch Änderungen durch Dritte während der Kommunikation
 - Erlaubt die Authentifikation der Kommunikationspartner durch Zertifikate
 - Sichert die Privatheit der Kommunikation durch Verschlüsselung
- Bei Errichtung einer SSL-Verbindung werden Verschlüsselungsmethoden ausgehandelt und Zertifikate überprüft

SSL in Java

- Implementierungen für SSL-Sockets im Paket [javax.net.ssl](#)
- SSL-Sockets werden dort durch Factories (Fabriken) erzeugt (statt durch Konstruktoren)
- Clientseitige Sockets:
 - Fabrik durch [javax.net.ssl.SSLSocketFactory.getDefault\(\)](#) ermitteln
 - Dort mit [createSocket](#) einen Socket erzeugen
- Serverseitige Sockets:
 - Fabrik durch [javax.net.ssl.SSLServerSocketFactory.getDefault\(\)](#) ermitteln
 - Dort mit [createServerSocket](#) einen Socket erzeugen
- Gelieferte [SSLSocket](#) und [SSLServerSocket](#) sind Unterklassen von [Socket](#) und [ServerSocket](#) mit SSL-Erweiterungen

Client für HTTP über SSL

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;

public class HttpClient {

    public static void main(String[] argv) {
        SSLSocket socket; // Nur, wenn man die Eigenschaften von SSLSocket nutzt

        try {
            SSLSocketFactory sslFactory =
                (SSLSocketFactory) SSLSocketFactory.getDefault();
            socket = (SSLSocket)sslFactory.createSocket(argv[0], 443);
        } catch (IOException iOExc) {
            System.err.println("Problem bei der Verbindungsaufnahme\n"+
                iOExc.getMessage());
            return;
        }
    }
}
```

Client für HTTP über SSL

```
try {
    OutputStream os = socket.getOutputStream();
    PrintWriter pw = new PrintWriter(os);
    // pw.println("GET / HTTP/1.0\n");
    pw.println("GET / HTTP/1.1\nHost: " + argv[0] + "\n");
    pw.flush();
    InputStream is = socket.getInputStream();
    BufferedReader br = new BufferedReader(new InputStreamReader(is));
    while (true) {
        String l = br.readLine();
        if (l == null) {
            break;
        } else {
            System.out.println(l);
        }
    }
} catch (IOException iOExc) {
    System.err.println("Problem beim Lesen\n" + iOExc.getMessage());
    return;
}
}
```



Zusammenfassung

Zusammenfassung

- HTTP
 - Anfrage/Antwort Interaktion
- Anfragen
 - Methoden
 - Anfrage- und Inhaltskopfzeilen
- Antworten
 - Antwortcodes
 - Antwortkopfzeilen
- HTTPS
 - SSL