



## ***Netzprogrammierung Hypertext Transfer Protocol HTTP***

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

# Überblick

- HTTP
- Anfragen
- Kopfzeilen
- Inhaltstypen
- Antworten
- Kopfzeilen
- HTTPS

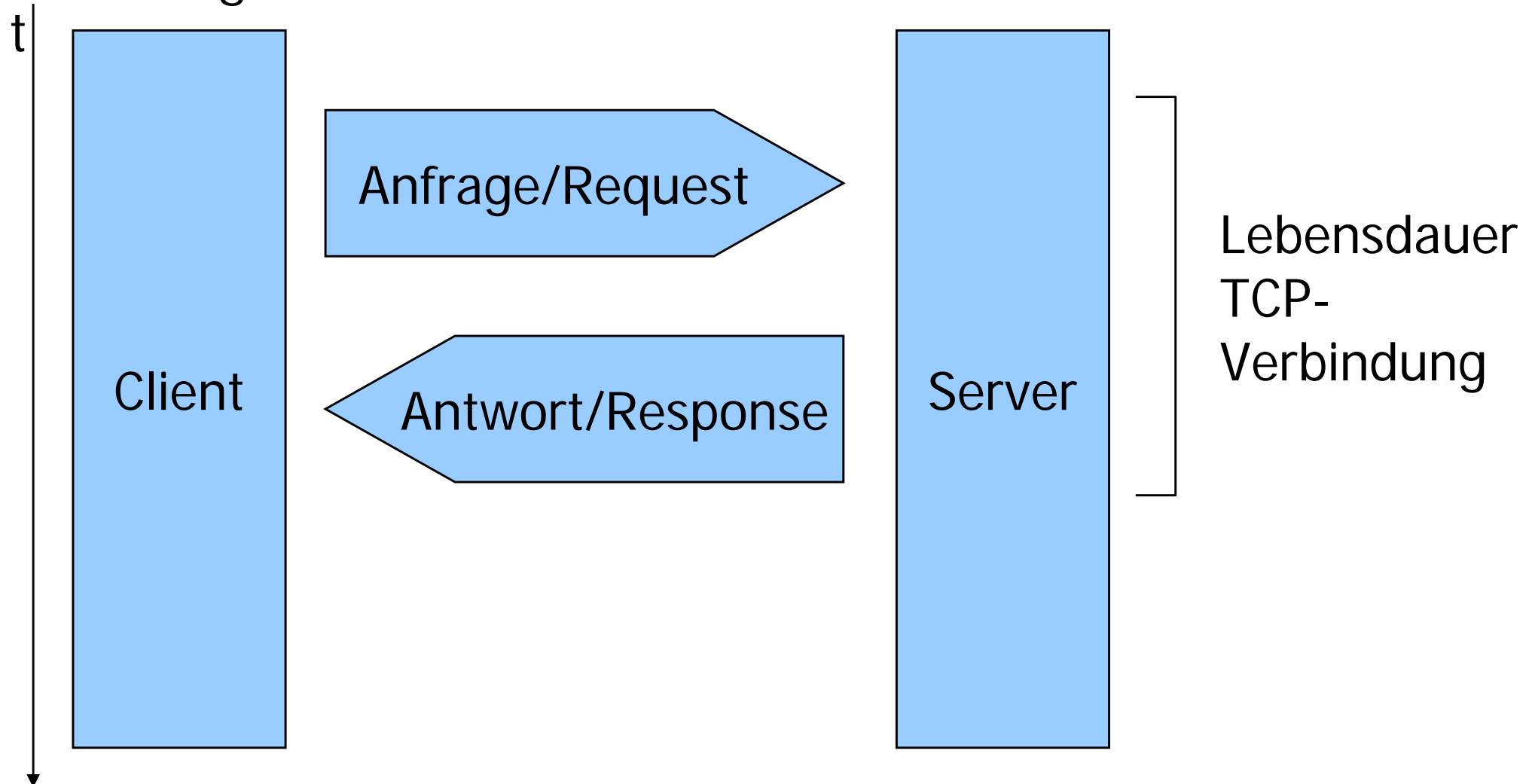


## **Hypertext Transfer Protocol HTTP (Überblick)**

# 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

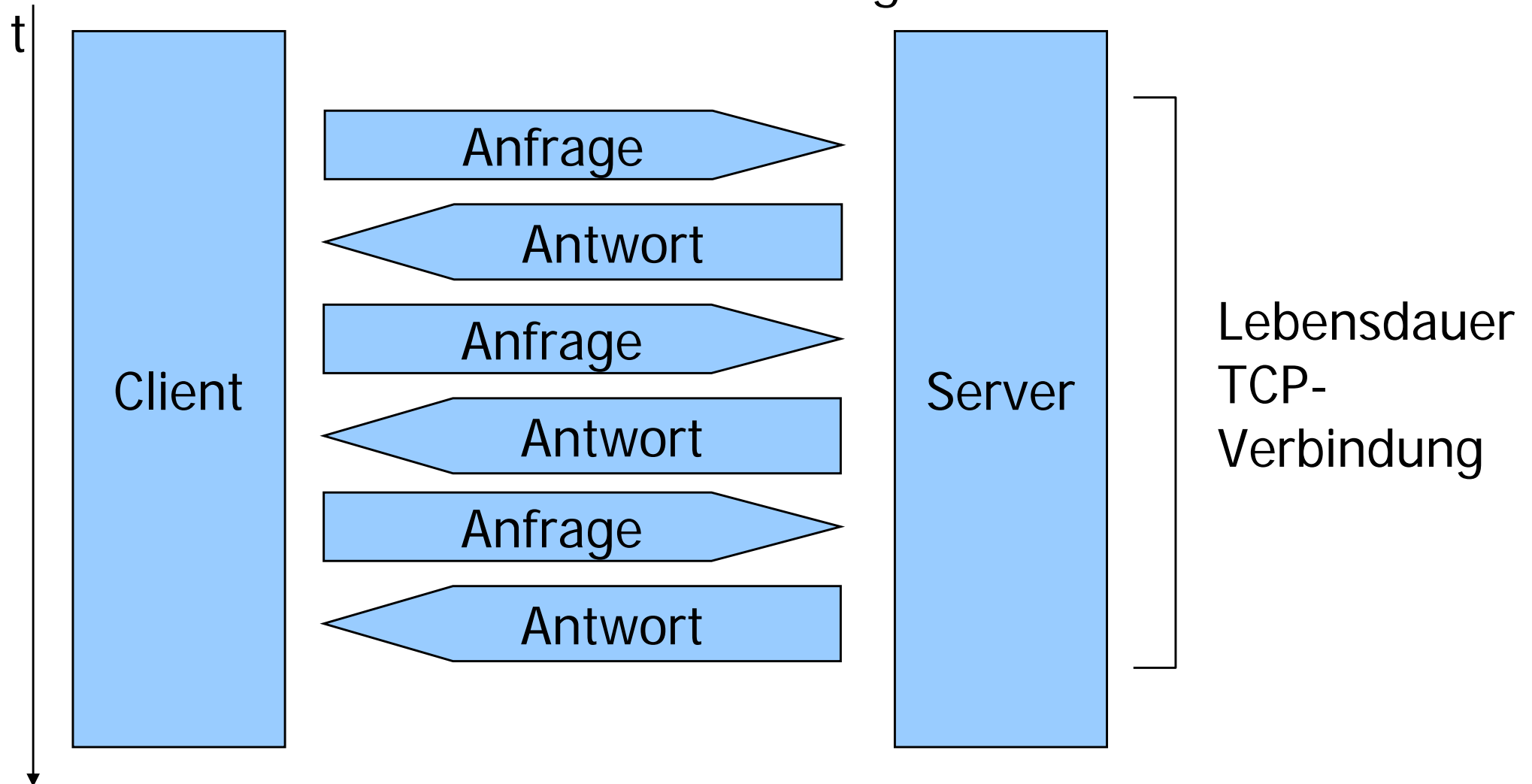
```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.04Gold (Win95; I)
Host: megababe.isdn:80
Accept: image/gif, image/jpeg, image/pjpeg, */*
```

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

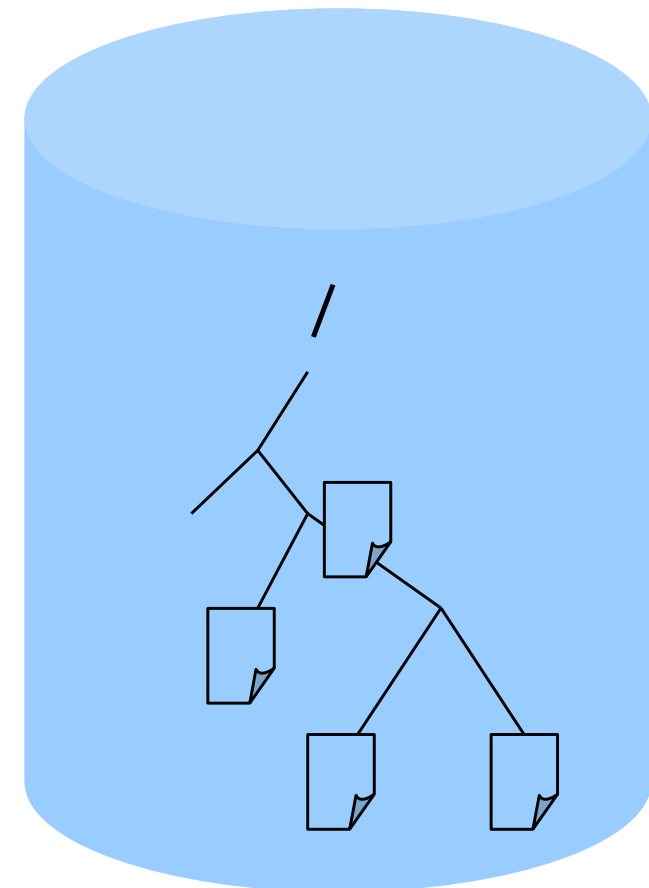
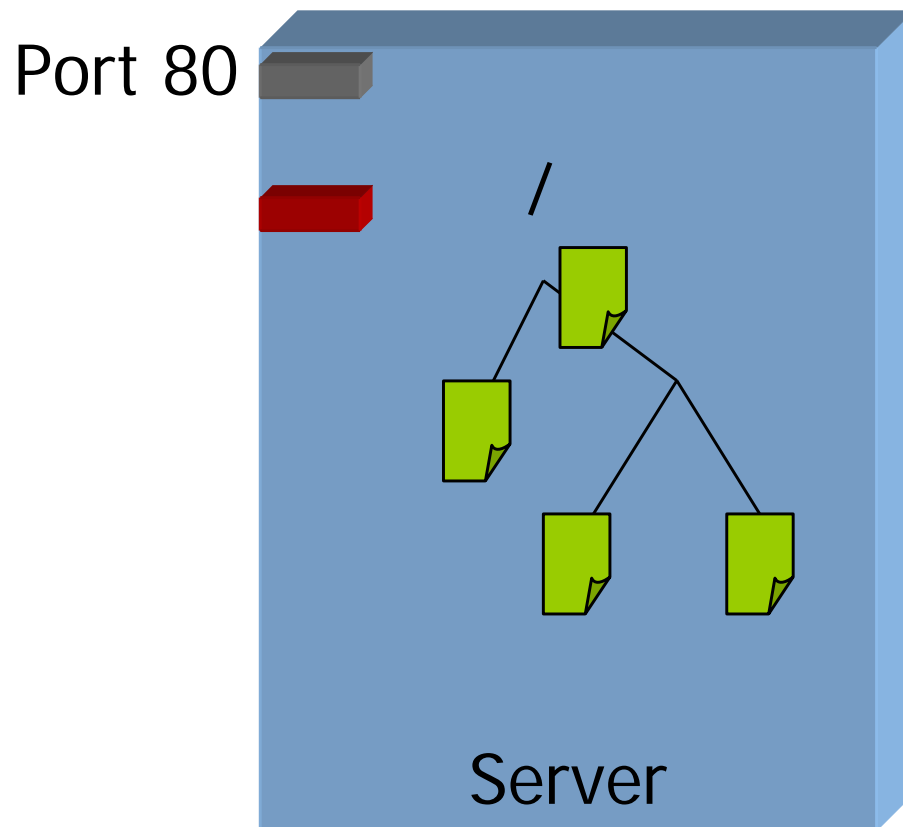
<HTML><HEAD><TITLE>TU Berlin ---
```

- 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

# HTTP Client

---

- 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;
        }
    }
}
```

# HttpClient/2

```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    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](http://www.w3.org)
    - HEAD <http://www.w3.org/Style/CSS/> HTTP/1.1 an Proxy [http-proxy.fu-berlin.de](http://http-proxy.fu-berlin.de)
  - Beantwortet mit Code, Kopfzeilen



# HttpHeadClient/1

```
try {
    PrintWriter pw = new PrintWriter(socket.getOutputStream());
    pw.println("HEAD / HTTP/1.0\n");
    // 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());
}
}
```

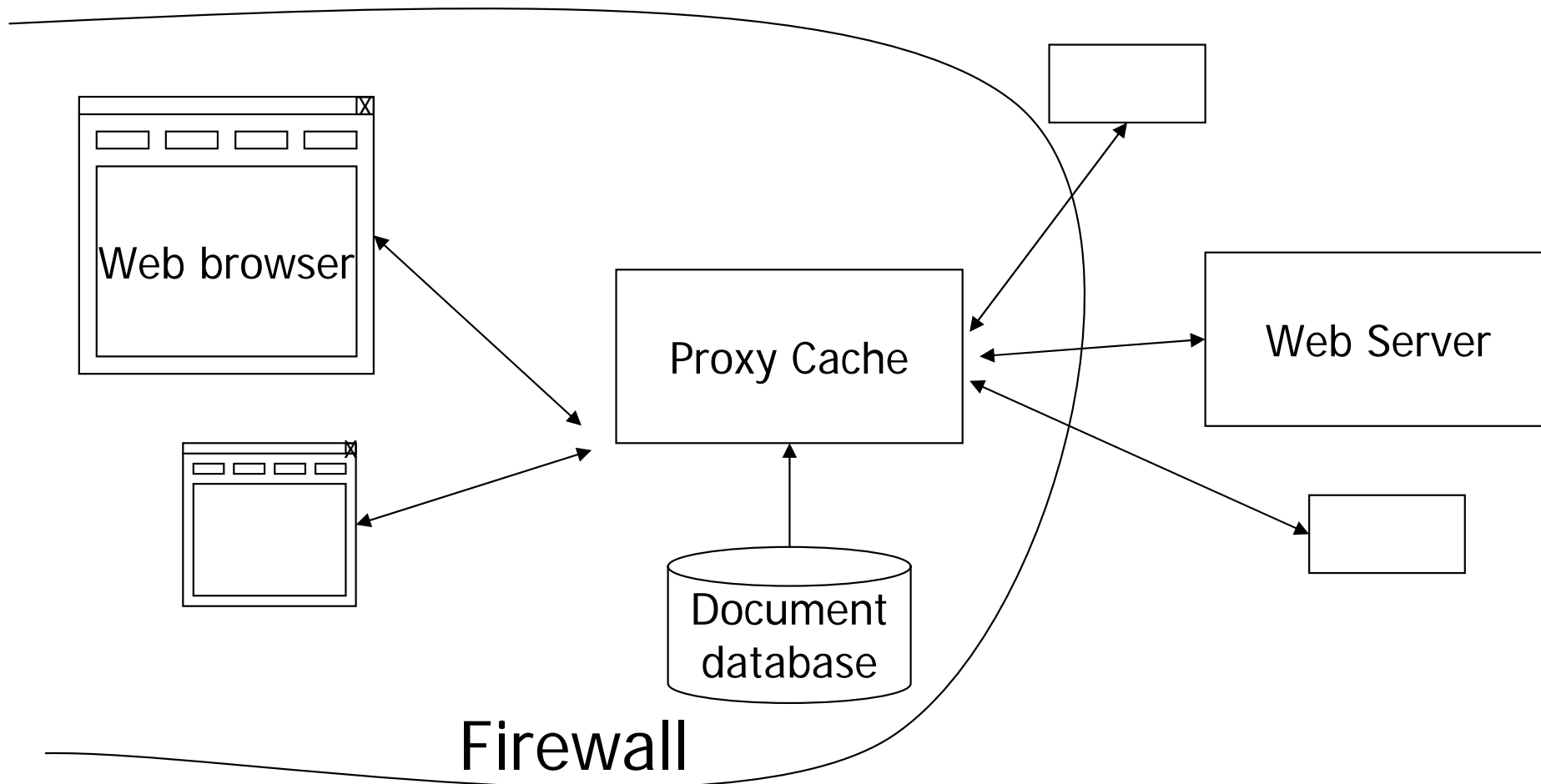
## HttpHeadClient/2

---

```
>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](http://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](http://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](http://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>
```



# Weiter 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



## Allgemeine und Anfrage-Kopfzeilen

- Date: Tue, 15 Nov 1994 08:12:31 GMT  
Datum des Abschickens der Anfrage im RFC 1123 Format
- Connection: close  
Verbindung nach Ergebnisübermittlung abbauen
- 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

- Transfer-Encoding: *Encoding*  
Wie die Mitteilung für den Transfer kodiert wurde
  - chunked: Mitteilung in Teilen geschickt, Zeichenanzahl in initialer Hexzahl

```
>java HttpClient11 focus.msn.de
java HttpClient11 focus.msn.de
HTTP/1.1 200 OK
Date: Fri, 25 Nov 2005 13:20:01 GMT
Server: Apache
set-cookie: NGUserID=11329248012594; path=/; domain=.msn.de;
  expires=fri, 10-aug-2012 16:48:59 gmt
Transfer-Encoding: chunked
Content-Type: text/html

2e96
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
  <html> <head>
  <title>FOCUS Online in Kooperation mit MSN Homepage</title>
  <meta
```
  - identity: Mitteilung unkodiert geschickt
  - gzip, compress, deflate: Komprimierte Übertragung

- *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

# Anfrage Kopfzeilen

---

- Host: *Name*  
Aus der URL ermittelter Name des Rechners von dem angefordert wird. *Einziger Pflichtkopfzeile in HTTP 1.1*
- 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;
        }
    }
}
```

```
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

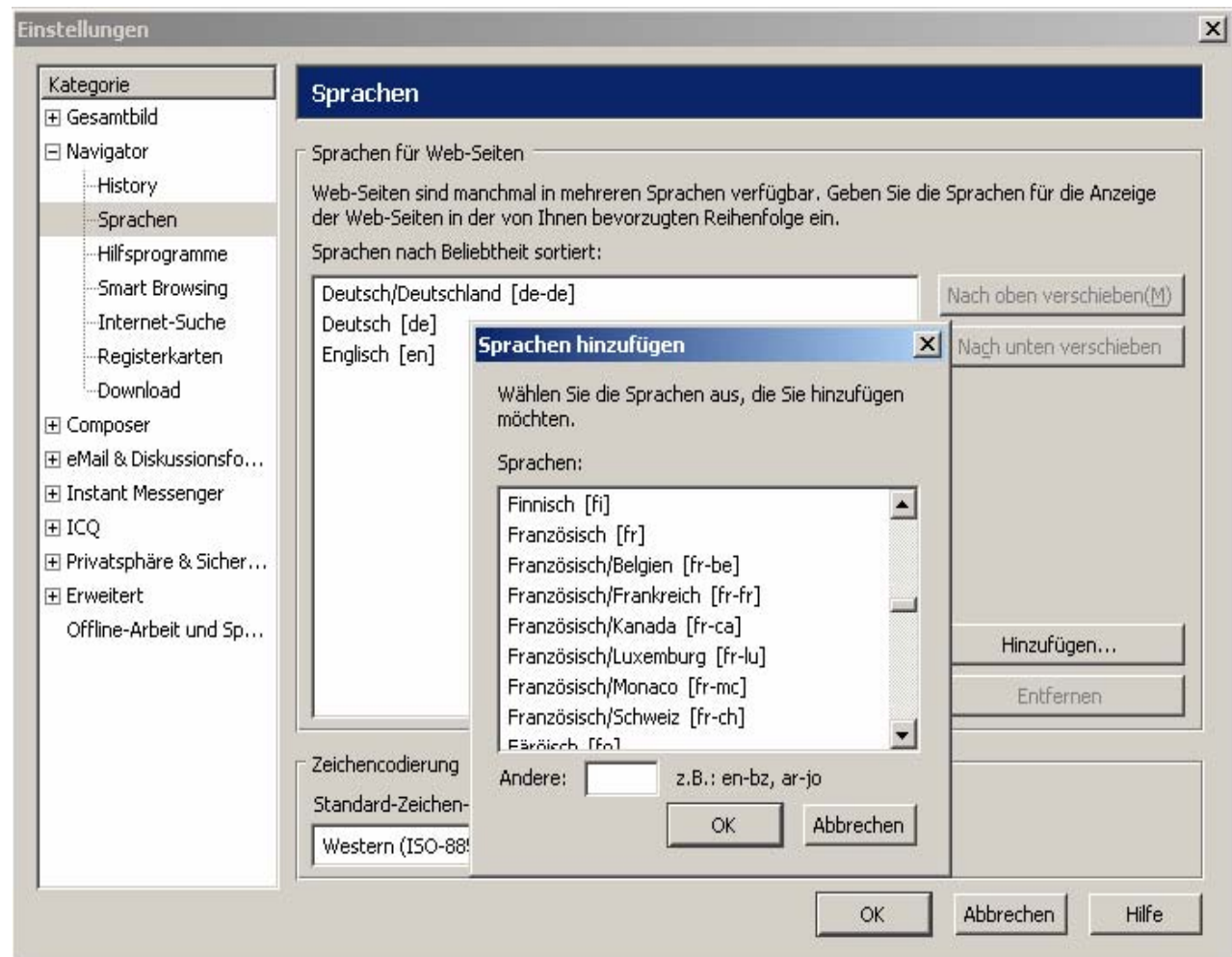
- Max-Forwards: *Anzahl*  
Wie oft ein OPTIONS oder TRACE weitergeleitet werden darf
- Range: *Bytebereich*  
Nur Teile der Information anfordern, Antwort ist dann 216  
Range: bytes=500-999
- Expect: *Token*  
Client erwartet bestimmte Eigenschaften von Server/Proxy  
(Falls nicht: 417)

# Anfrage Kopfzeilen

- From: *Mailadresse*  
Nutzer
- User-Agent: *Produkt/Version*  
Browser z.B.  
(Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
- Referer: *URL*  
Seite auf der ein Link auf die angeforderte Seite stand
- 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

# Sprache in HTTP

- Browser kann Präferenzen im HTTP-Request mitteilen:  
GET / HTTP/1.1  
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



# Zeicheneigenschaften

- 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									0	280	281	282	283	284	285	286	287
	⠠⠠	⠠⠠									1	2801	2811	2821	2831	2841	2851	2861	2871
	⠠⠠	⠠⠠									2	2802	2812	2822	2832	2842	2852	2862	2872
	28B4	28C4									3	2803	2813	2823	2833	2843	2853	2863	2873
	⠠⠠	⠠⠠									4	2804	2814	2824	2834	2844	2854	2864	2874
	⠠⠠	⠠⠠									5	2805	2815	2825	2835	2845	2855	2865	2875
	28B5	28C5									6	2806	2816	2826	2836	2846	2856	2866	2876
	⠠⠠	⠠⠠									7	2807	2817	2827	2837	2847	2857	2867	2877
	⠠⠠	⠠⠠									8	2808	2818	2828	2838	2848	2858	2868	2878
	28B6	28C6									9	2809	2819	2829	2839	2849	2859	2869	2879
	⠠⠠	⠠⠠									A	280A	281A	282A	283A	284A	285A	286A	287A
	⠠⠠	⠠⠠									B	280B	281B	282B	283B	284B	285B	286B	287B
	28B7	28C7									C	280C	281C	282C	283C	284C	285C	286C	287C
	⠠⠠	⠠⠠									D	280D	281D	282D	283D	284D	285D	286D	287D
	⠠⠠	⠠⠠									E	280E	281E	282E	283E	284E	285E	286E	287E
	28B8	28C8									F	280F	281F	282F	283F	284F	285F	286F	287F



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 ི TIBETAN MARK CHE MGO	0F5E ཉ TIBETAN LETTER ZHA	
0F39 ◌ TIBETAN MARK TSA -PHRU • a lenition mark	0F5F ཐ TIBETAN LETTER ZA	
	0F60 ད TIBETAN LETTER -A	
	0F61 དྷ TIBETAN LETTER YA	
	0F62 ན TIBETAN LETTER RA • when followed by a subjoined letter = ra mgo	
<b>Paired punctuation</b>	0F63 པ TIBETAN LETTER LA	
0F3A ཏྟ TIBETAN MARK GUG RTAGS GYON	0F64 ཕ TIBETAN LETTER SHA	
0F3B ཏྟ TIBETAN MARK GUG RTAGS GYAS • brackets	0F65 བ TIBETAN LETTER SSA = reversed sha	
0F3C } TIBETAN MARK ANG KHANG GYON	0F66 མ TIBETAN LETTER SA	
0F3D } TIBETAN MARK ANG KHANG GYAS • used for bracketing with a roof over	0F67 ཙ TIBETAN LETTER HA	
	0F68 ཚ TIBETAN LETTER A • base for dependent vowels	
	0F69 ཛ TIBETAN LETTER KSSA ≡ 0F40 ཏྟ 0FB5 ཏྟ	
	0F6A ཛ TIBETAN LETTER FIXED-FORM RA • used only in transliteration and transcription	
<b>Astrological signs</b>		
0F3E ཏྟ TIBETAN SIGN YAR TSHES		
0F3F ཏྟ TIBETAN SIGN MAR TSHES • marks which combine with digits		
<b>Consonants</b>	<b>Dependent vowel signs</b>	
0F40 ཏྟ TIBETAN LETTER KA	0F71 ཏྟ TIBETAN VOWEL SIGN AA = a-chung • common, vowel-lengthening mark	
0F41 ཏྟ TIBETAN LETTER KHA	0F72 ཏྟ TIBETAN VOWEL SIGN I	
0F42 ཏྟ TIBETAN LETTER GA	0F73 ཏྟ TIBETAN VOWEL SIGN II • use of this character is discouraged	
0F43 ཏྟ TIBETAN LETTER GHA ≡ 0F42 ཏྟ 0FB7 ཏྟ		
0F44 ཏྟ TIBETAN LETTER NGA		
0F45 ཏྟ 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“
  - meint Encoding!

# Anfrage Kopfzeilen

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

# Anfrage mit Priorität

- 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

# 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
- Kann auf Transportweg (Proxies) geändert werden

# Anfrage Kopfzeilen

- Accept-Encoding: *Kodierung*

Binäre Daten	binary	Inhaltscodierung
8-Bit-Daten	8bit	
7-Bit-Daten	7bit	
uuencode-kodiert	quoted-printable	
base64-kodiert	base64	
...		
komprimiert	gzip, compress, deflate	Transfer-codierung
in Teilen	chunked	
unkodiert	identity	

- Accept-Language: *Sprachkürzel*
  - Accept-Language: de, en

# 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



## **Inhaltskopfzeilen bei Anfrage und Antwort**



# Inhalts-Kopfzeilen

---

- Content-Encoding: *Kodierung*  
Kodierung des Inhalts
  - binary, 8bit, 7bit, quoted-printable, base64, ...
- Content-Transfer-Encoding: *Kodierung*  
Transferkodierung
  - gzip, compress, deflate, chunked, identity
- 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*  
Verweis auf eigentlichen Inhalt
- Content-MD5: *MD5Checksum*  
Message Digest für Inhalt zur Integritätsprüfung
- Expires: *Datum*  
Kann nach Datum aus Caches gelöscht werden
- Last-Modified: *Datum*  
Letzte Änderung



## Inhaltstypen / MIME

# 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



## 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...




Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://luftbruecke.inf.fu-berlin.de/

\*\*\* 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.



luftbruecke (FunkLAN) Zugang - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

<https://luftbruecke.inf.fu-berlin.de/>



**Willkommen auf Luftbruecke!**

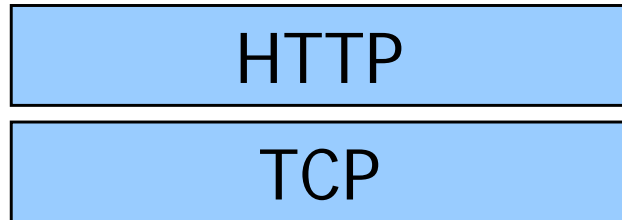
**Welcome at Luftbruecke!**  
 ( luftbruecke = airlift )

Neues Wireless-Passwort generieren:  
 Generate a new wireless password:

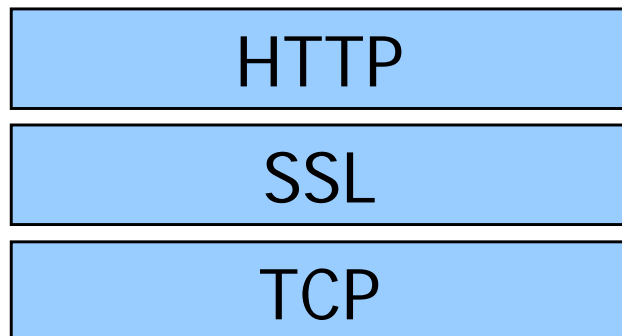


# HTTP über SSL Sockets

- 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

- 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
  - 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
- Inhaltstypen
  - MIME
- Antworten
  - Antwortcodes
  - Antwortkopfzeilen
- HTTPS
  - SSL