
Internet-Technologien

Sommersemester 2004

1. Internet-Technologien

Sommersemester 2004

Dr. Heinz Kredel

Rechenzentrum Universität Mannheim

- 4 Stunden Vorlesung und 2 Stunden Übungen
- Klausurtermin: 1. Oktober 2004
- Zwischenfragen sind willkommen

Lernziele

- Einführung in Konzepte, Theorien und Techniken der Internet- und Web-Programmierung sowie dem Design und der Entwicklung von Internet- und Web-Anwendungen.
- Übungen mit ausgewählter Software

URL

- <http://krum.rz.uni-mannheim.de/inet/>

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Sun Aug 8 21:52:12 CEST 2004

Inhaltsverzeichnis

1. [Internet-Technologien](#)
2. [Einleitung](#)
3. [Internet und TCP/IP](#)
4. [Java - Sprache](#)
5. [Java Sockets](#)
6. [Extensible Markup Language \(XML\)](#)
7. [Hypertext Markup Language \(HTML\)](#)
8. [Extensible Hypertext Markup Language \(XHTML\)](#)
9. [Cascading Style Sheets \(CSS\)](#)
10. [Cascading Style Sheets 2 \(CSS2\)](#)
11. [JavaScript](#)
12. [Document Object Model \(DOM\)](#)
13. [Namespaces und XLink](#)
14. [XPath und XPointer](#)
15. [XML Style Sheets \(XSL\)](#)
16. [XSL Formatting Objects](#)
17. [XML Schema](#)
18. [Wireless Markup Language \(WML\)](#)
19. [Java API for XML Processing \(JAXP\)](#)
20. [Metadaten: DC und RDF](#)
21. [Email](#)
22. [PGP und Kryptographie](#)
23. [SSL und TLS](#)
24. [Gnutella](#)
25. [JXTA](#)
26. [P2P Suche](#)

- 27. [Hypertext Transfer Protocol \(HTTP\)](#)
- 28. [Interaktion und CGI](#)
- 29. [Perl und CGI](#)
- 30. [PHP Hypertext Preprocessor \(PHP\)](#)
- 31. [Web-Datenbanken: MySQL](#)
- 32. [HTTP over SSL \(HTTPS\)](#)
- 33. [Java - Applets und Servlets](#)
- 34. [SOAP & WSDL](#)
- 35. [Apache Axis](#)
- 36. [Web-CMS](#)
- 37. [e-learning](#)
- 38. [Audio & Video](#)
- 39. [Schlussbemerkungen](#)
- 40. [Literatur](#)

2. Einleitung

Das Thema *Internet* hat betriebswirtschaftliche, künstlerische, gesellschaftliche und *technische* Aspekte.
Schlagworte: e-commerce, m-commerce.

- Hardwareaspekte
- Softwareaspekte
- Sitekonzeption
- Konzeption als Informationssystem

Hardwareaspekte

- Elektrotechnik des Internet
- Computertechnik
- Konzeption (grosser) Web-Sites
- welche / wieviele Netzkomponenten ?
Switches, Leitungen
- welche / wieviele Web-Server ?
- welche / wieviele Caches, Proxys, Firewalls ?

Softwareaspekte

- Kommunikationsprotokolle: TCP/IP, WAP
- Anwendungsprotokolle: HTTP, SMTP
- Protokolle zur Sicherheit: SSL, TLS, PGP
- Auszeichnungs-/Markierungs-Sprachen: HTML, XHTML, WML
- Meta-Sprachen: XML, Schema
- Programmiersprachen: JavaScript, Perl, Java, PHP
- Formatierungssprachen: CSS, XSL
- Datenbanken: MySQL, Oracle
- Web-Server Software: Apache
- Web-Client Software: Browser, User Agents

Sitekonzeption

- Grafische Gestaltung
- Benutzeroberfläche
- Benutzerführung
- Realisierbarkeit / Programmierbarkeit

Konzeption von Informationssystemen

- Zielgruppe
- Angebot
Warenkatalog, Kaufhaus, Informationszentrum, Koordinierungsinstrument, Verwaltungsinstrument

- Informationsflüsse in und zwischen DV-Systemen
 - Datenbank Einsatz und Einbindung
 - Organisation der Füllung / Aktualisierung
 - ERP Einsatz und Einbindung
SAP R/3 (mySAP.com)
-

2.1. Überblick

Orientierung an der Arbeit des World Wide Web Consortiums [W3C](#) und der Internet Engineering Task Force [IETF](#)

Abgrenzung

- Software Orientierung
- Seminar <==> Vorlesung
- Übungen
- keine Designfragen
- keine Site-Konzeption
- kein Marketing-Konzeption
- keine Hardwareaspekte

Client und Server



- Internet und TCP/IP
- Java Programmierung, Sockets, Applets und Servlets
- Extended Markup Language (XML)
- Hypertext Markup Language (HTML) 3.2 und 4.0
- Extensible Hypertext Markup Language (XHTML) 1.0
- Cascading Style Sheets (CSS), CSS1, CSS2, DSSSL
- JavaScript, ECMA-Script
- Document Object Model (DOM)
- XML Style Sheets (XSL)
- XML-Schema
- WAP und Wireless Markup Language (WML)
- Java API for XML Processing (JAXP)
- Metadaten, Dublin Core, RDF Resource Description Framework
- Simple Mail Transfer Protokoll (SMTP)
- Sicherheit: Secure Socket Layer (SSL) 3.0, Transport Layer Security (TLS) 1.0, HTTPS
- Peer-to-peer Protokolle: Gnutella, (Java) JXTA
- Hypertext Transfer Protokoll (HTTP) und Web-Server Apache
- Simple Object Access Protocol (SOAP)

- Common Gateway Interface (CGI)
- Perl Programmierung, CGI Modul
- Server Side Includes, PHP Hypertext Preprocessor
- Datenbank Zugriff via Web, MySQL
- Web Content Management Systems (Web-CMS): Zope, PHP-Nuke, IONAS
- Web-Groupware: Basic Support for Cooperative Work (BSCW)
- Privatsphäre, Platform for Privacy Preferences (P3P) Project
- Proxies und Caches, Squid

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:43:01 CEST 2004

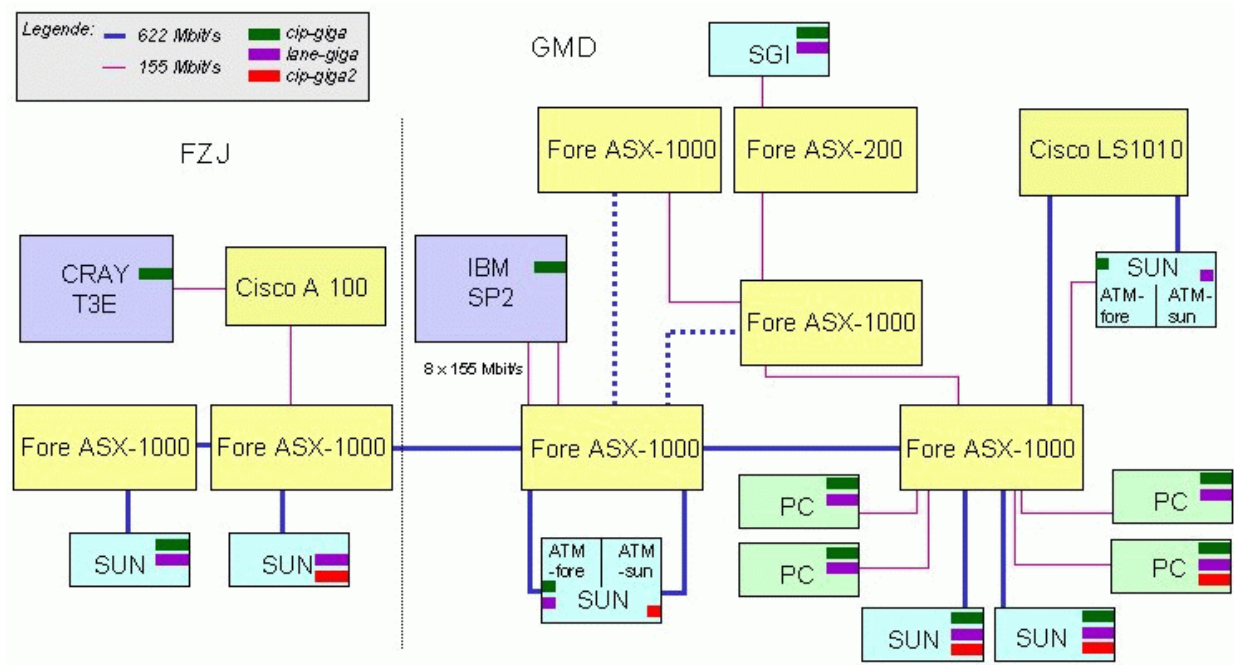
3. Internet und TCP/IP

- Internet als Netz von Netzen
- TCP/IP
- Anwendungen

Deutschland, Welt



Gigabit Testbed



Quelle: DFN e.V.

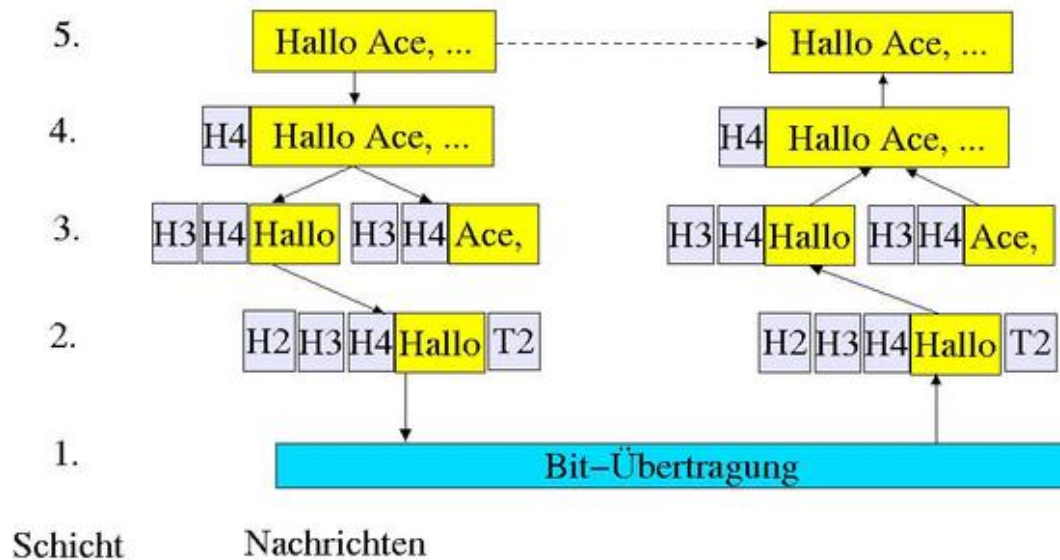
3.1. TCP/IP

Protokoll-Schichten

Schicht	Bemerkungen
5. Verarbeitung	Telnet, FTP, SMTP, NNTP, HTTP, SSH
4. Transport	TCP, UDP
3. Vermittlung	IP, Internet Protocoll
2. Sicherung	Adapter-Karten
1. Bitübertragung	Leitungen, Elektronik

nach Tanenbaum

Informationsfluss in Protokoll Schichten



- IP-Pakete: Transport der Informationen in Paketen
- IP-Adresse: 134.155.50.127

```
% ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
        UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
        RX packets:28 errors:0 dropped:0 overruns:0
        TX packets:28 errors:0 dropped:0 overruns:0

dummy0  Link encap:10Mbps Ethernet  HWaddr 00:00:00:00:00:00
        inet addr:10.1.1.254  Bcast:10.255.255.255  Mask:255.0.0.0
        UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0
        TX packets:0 errors:0 dropped:0 overruns:0

ippp0   Link encap:Point-Point Protocol
        inet addr:134.155.18.49  P-t-P:134.155.52.241  Mask:255.0.0.0
        UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
        RX packets:3711 errors:0 dropped:0 overruns:0
        TX packets:4562 errors:0 dropped:0 overruns:0

eth0    Link encap:10Mbps Ethernet  HWaddr 00:00:C0:F7:8B:2D
        inet addr:10.1.1.254  Bcast:10.1.1.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:1414 errors:0 dropped:0 overruns:0
        TX packets:2157 errors:0 dropped:0 overruns:0
        Interrupt:10 Base address:0x310 Memory:d4000-d8000
```

- Verbindungstest

```
% ping 134.155.50.127
PING 134.155.50.127: 56 data bytes
64 bytes from 134.155.50.127: icmp_seq=0. time=63. ms
64 bytes from 134.155.50.127: icmp_seq=1. time=62. ms
64 bytes from 134.155.50.127: icmp_seq=2. time=63. ms
64 bytes from 134.155.50.127: icmp_seq=3. time=31. ms
64 bytes from 134.155.50.127: icmp_seq=4. time=31. ms

----134.155.50.127 PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 31/50/63
```

```
% netstat -i
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flags
lo 3584 0 28 0 0 0 28 0 0 0 BLRU
dummy 1500 0 0 0 0 0 0 0 0 0 BORU
ippp0 1500 0 3711 0 0 0 4562 0 0 0 OPRU
eth0 1500 0 1414 0 0 0 2157 0 0 0 BRU
```

- Domainnamen: rechner.subnetz.country.area

```
% nslookup www.netscape.com
Server: rumms.uni-mannheim.de
Address: 134.155.50.52

Non-authoritative answer:
Name: www3.netscape.com
Address: 205.218.156.44
Aliases: www.netscape.com
```

- Routing: Wege der IP-Pakete

```
% traceroute www.netscape.com

0 hicomgw1s.uni-mannheim.de (134.155.30.254) 31 ms 63 ms 31 ms
1 hicomgw1s.uni-mannheim.de (134.155.30.254) 31 ms 32 ms 62 ms
2 manhattan.uni-mannheim.de (134.155.50.200) 31 ms 62 ms 32 ms
3 Mannheim1.BelWue.DE (129.143.61.1) 31 ms 31 ms 62 ms
4 Uni-Mannheim1.WiN-IP.DFN.DE (188.1.5.37) 32 ms 63 ms 62 ms
5 ZR-Karlsruhe1.WiN-IP.DFN.DE (188.1.5.33) 32 ms 62 ms 31 ms
6 ZR-Frankfurt1.WiN-IP.DFN.DE (188.1.144.37) 63 ms 31 ms 63 ms
7 IR-Perryman1.WiN-IP.DFN.DE (188.1.144.78) 156 ms 156 ms *
8 166.48.39.253 (166.48.39.253) 156 ms 156 ms 125 ms
9 core2.SanFrancisco.mci.net (204.70.4.201) 219 ms 219 ms 250 ms
10 borderx2-fddi-1.SanFrancisco.mci.net (204.70.158.68) 219 ms 219 ms 219 ms
11 netscape-ds3.SanFrancisco.mci.net (204.70.158.122) 218 ms 219 ms 218 ms
12 h-207-200-71-20.netscape.com (207.200.71.20) 219 ms 219 ms 218 ms
13 www24.netscape.com (207.200.73.73) 219 ms * 219 ms
```

```
% netstat -r
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
kredel-wh.isdn. * 255.255.255.255 UH 1500 0 0 dummy0
tc1.rz.uni-mann * 255.255.255.255 UH 1500 0 0 ippp0
10.1.1.0 * 255.255.255.0 U 1500 0 0 eth0
loopback * 255.0.0.0 U 3584 0 0 lo
default * 0.0.0.0 U 1500 0 0 ippp0
```

- Beispiele:

```
ifconfig lo 127.0.0.1
route add localhost 127.0.0.1 0
route add default ippp0
```

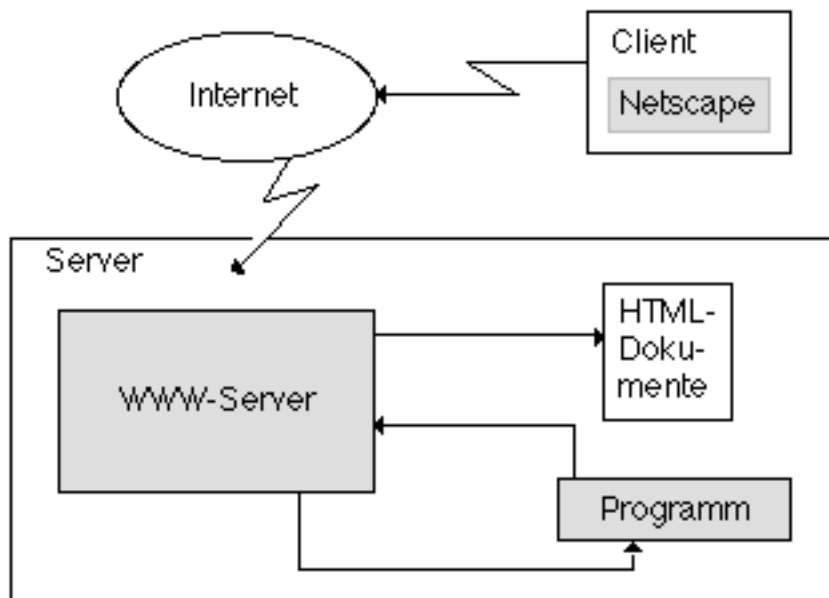
- TCP/IP unter DOS und Windows95/NT:
- ping
- netstat
- route
- kein ifconfig, traceroute
Ersatz: winipcfg, ipconfig, tracert
- Netzkonfiguration
IP-Adresse, Gateway, DNS-Server

Zugang zum Internet

- Provider bieten Verbindung zum Internet
Online-Dienste und Server-Betreiber
 - Dial-in:
Telefonanschluß, Modem,
Terminalprogramm, Decoder
 - Dial-up-IP: eigene IP-Adresse
SLIP Serial Line IP,
PPP Point to Point Protokoll
Browser, FTP
 - Standleitungen:
LAN Local Area Network, "Intranet"
WAN Wide Area Network
 - Kostenteilungsprinzip:
 1. Zahlen des eigenen Netzes (LAN, Karten)
 2. Zahlen von Verbindungen zu einigen Nachbarn (WAN, Modem)
 3. Alle Welt darf meine Leitungen benutzen
 4. Ich habe das Recht in aller Welt Leitungen zu benutzen
-

3.2. Anwendungen

Client-Server Prinzip



Verschiedene TCP/IP Anwendungen auf einem Server bzw. Client werden durch die **Port-Nummer** unterschieden.

Die Port-Nummer ist eine Zahl zwischen 0 und 65535 ($=2^{16}-1$).

Die Port-Nummern zwischen 0 und 1023 sind nur von Root / Administrator verwendbar.

Die Zuordnung wird von [IANA](#) (Internet Assigned Numbers Authority) verwaltet.

Die Zuordnung von Anwendung zu Port-Nummer steht in der Datei `/etc/services`.

Bekannte Ports sind 80 (HTTP), 22 (SSH) oder 25 (SMTP).

```
ftp      21/tcp      # File Transfer [Control]
ssh      22/tcp      # SSH Remote Login Protocol
telnet   23/tcp      # Telnet
smtp     25/tcp      mail # Simple Mail Transfer
www      80/tcp      # World Wide Web HTTP
sftp     115/tcp     # Simple File Transfer Protocol
```

WWW

Uni Mannheim

- Graphische Oberfläche
Browser: Mosaic, Netscape, MS Internet Explorer, Amaya, HotJava
- Textmode Oberfläche
Browser: Lynx, Perl basierte

Telnet

Rummelplatz, Localhost

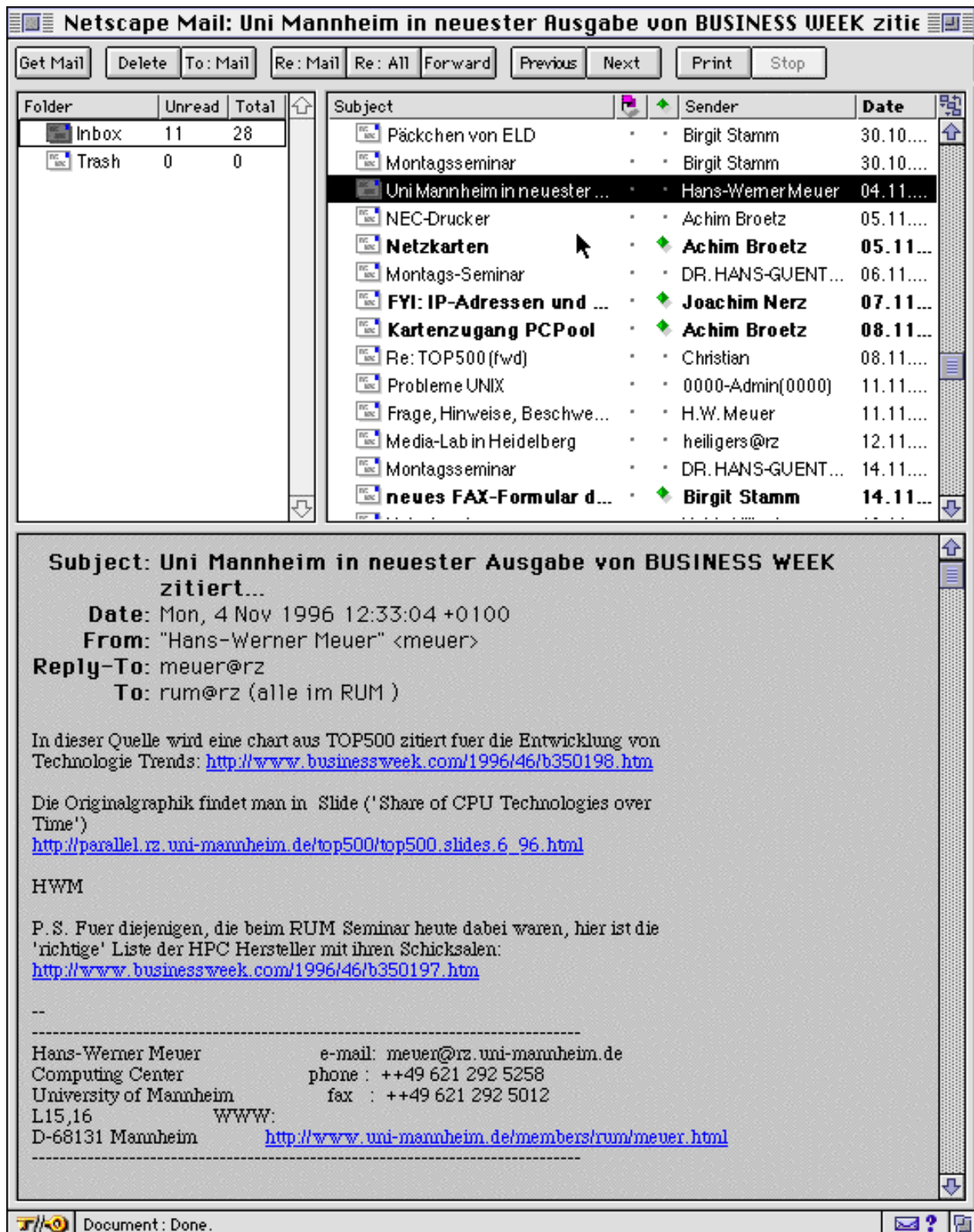
Heute durch SSH (Secure Shell) ersetzt (teraterm, putty, ssh), das verschlüsselte Datenübertragung ermöglicht.

FTP

FTP, Localhost

Heute durch SCP (Secure Copy) ersetzt (WinSCP, SFTP), das verschlüsselte Datenübertragung ermöglicht.

Email



mailto:kredel@rz.uni-mannheim.de

News

Netscape News: GATEWAY FAQ: Fragen & Antworten zur Mail-Adress

To: News To: Mail Re: Mail Re: News Re: Both Forward Previous Next Thread Group

News Server

- de.* (282 groups)
 - de.admin.* (11 groups)
 - de.admin.archiv
 - de.admin.lists
 - de.admin.mail
 - de.admin.misc
 - de.admin.news.* (6 groups)

Subject

- Re: beliebiger WWW-nichtPO...
- Re: UCE von lsp.inter.net -> (UCE...
- eRobinson (was: Re: UCE von Is...
- GATEWAY FAQ: Fragen & Antwor...**
- The curious reply of a "SPA...**
- Re: test

Sender

- Axel Wittr
- Robert B
- Kai Felt
- Ulf Moelle
- Robert
- Gideon

Subject: GATEWAY FAQ: Fragen & Antworten zur Mail-Adressierung
Date: Thu, 12 Dec 1996 10:24:24 GMT
From: Ulf_Moeller@public.uni-hamburg.de (Ulf Moeller)
Reply-To: um@c2.net (Ulf Moeller)
Organization: private site, Hamburg (Germany)
Newsgroups: [de.comm.gateways](#), [de.admin.mail](#), [de.answers](#), [news.answers](#)
Followup-To: [de.comm.gateways](#)

Archive-Name: de-gateways/faq
 URL: <http://www.c2.net/~um/faq/gateways.txt>
 Version: 1.8d

GATEWAY FAQ: Oft gestellte Fragen & Antworten zur Mail-Adressierung
 =====

Inhalt

- Domains
- Adressierung aus dem FidoNet, MausNet, CompuServe usw.
- Domain-Adressen der Netze
- Sonstiges

1. Domains...

=====

Um EMail für Millionen Teilnehmer auf der ganzen Welt verwalten zu können, benutzt man zur Adressierung organisatorische bzw. geographische Bereiche, die Domains.

Eine Domain-Adresse sieht zum Beispiel so aus: heinz@foo.sh.sub.de

Document: Done.

[news:tips-infos](#)

3.3. Ausblick

- Neuordnung der Namensvergabe, Namensräume
 - IPv6, IP version 6
 - Übermittlung von Multimedia Daten
 - WAP, i-Mode, mobile IP
 - Verschlüsselte Datenübertragung, SSL
-

© Universität Mannheim, Rechenzentrum, 1998-2004.

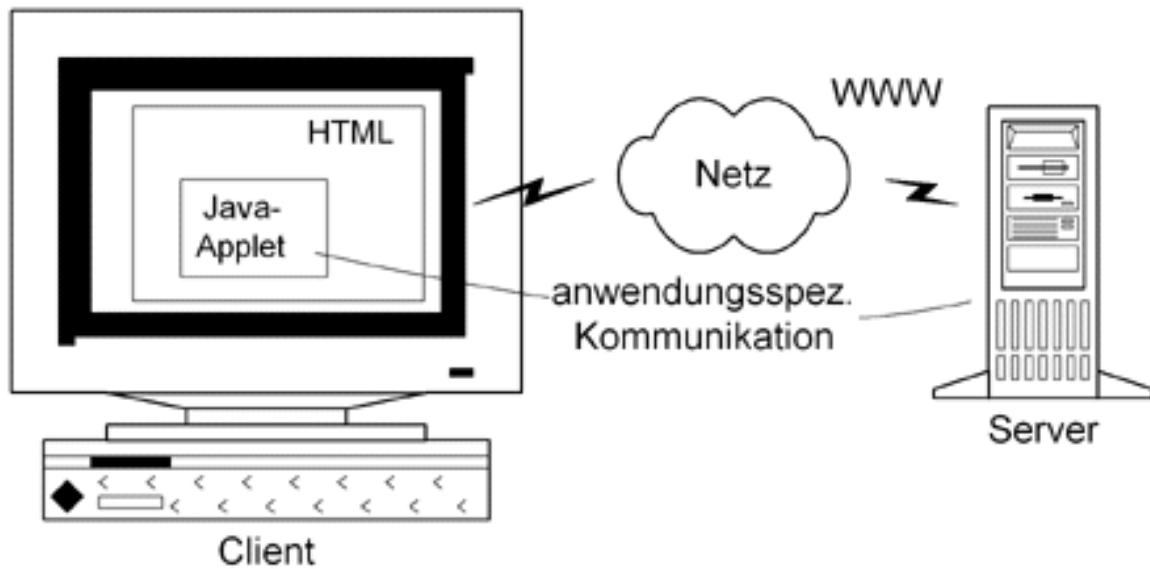
Last modified: Wed May 26 23:00:05 CEST 2004

4. Java - Sprache

- Einleitung
-

4.1. Einleitung

- Programmiersprache der Firma SUN
- vollständige objektorientierte Programmiersprache
- Syntaktisch an C++ angelehnt
- Objekte, Klassen, Kapselung
- Vererbung, Polymorphismus
- multi-threaded
- hierarchisches Modulkonzept package
- Interpreter
- Unterschiede zu anderen OO-Sprachen
- einfach, Verzicht auf wenig genutzte höhere Sprachkonzepte
- teilcompilierend, Byte-Code kann auf jeder Plattform mit einer Implementierung der zu Java gehörenden 'Virtual Machine' ablaufen
- dynamisches Laden von Klassen
- sehr sehr umfangreiche Bibliotheken als Teil der Sprachdefinition
- Sicherheit
- Java als "Web-Programmiersprache"
- Virtual Machine ist in vielen gängigen Web-Browsern verfügbar
- einfache Integration mit HTML
- Bibliotheken für graphische Oberflächenelemente (AWT) ist Teil der Laufzeit-Umgebung
- Bibliothek für Netzwerk-Programmierung ist Teil der Laufzeit-Umgebung
- dynamisches Laden von Klassen ermöglicht effiziente Nutzung von niedriger Netz-Bandbreite
- Mechanismen zur Unterscheidung zwischen "vertrauenswürdigen" und sogenannten "untrusted" Code sind Teil der Sprachdefinition und Laufzeitumgebung



Code-Umfang der JDKs

Abbildung:
Lines-of-Code der JDKs

was \ JDK Version	1.0.2	1.1.8	1.2.2	1.3.1	1.4.0
lines of Java code	36.080	152.347	502.726	574.034	1.183.279
words in Java code	156.799	707.193	2.085.638	2.373.953	4.718.321
bytes in src-dir	1.0 MB	7.0 MB	20.0 MB	24.0 MB	48.0 MB
public class or interface	122	736	1.649	1.952	3.993
directories / packages	14	35	88	98	238
@authors, see 1	21	114	100	236	315
if ()	1.319	4.648	18.209	23.190	44.006
for ()	236	1.692	2.727	5.065	10.031
while ()	105	269	985	1.134	1.625
switch ()	33	107	410	465	839
try { }	93	112	507	602	2.143
catch ()	108	307	909	1.126	3.505
throw new	167	709	2.011	2.294	6.700
boolean	322	1.029	4.286	4.818	9.532
int	1.556	5.043	19.613	21.599	41.679
long	130	810	754	1.890	3.250
float	43	104	1.030	1.134	1.625
double	71	181	1.057	1.427	1.862
public	1.976	5.828	21.110	24.368	50.209

private	303	1.948	5.120	6.477	13.900
protected	151	689	3.695	4.201	6.399
abstract class	17	74	214	319	545
<i>/** comments */</i>, see 2	2.268	6.503	22.443	25.492	49.592

Bemerkungen:

1. ohne Namen zu normalisieren und nur die genannten
2. d.h. öffentliche mit javadoc dokumentierte Objekte

Die Zahlen wurden im wesentlichen mit `grep pattern cat.java|wc` ermittelt. Die Datei `cat.java` wurde mit `find src-dir -name "*.java" |xargs cat > cat.java` erzeugt.

Daneben gibt es in den jeweiligen JDKs noch viele class-Dateien ohne java-Dateien, wo also der Source-Code nicht bekannt ist.

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Wed May 26 23:34:56 CEST 2004

5. Java Sockets

- Sockets
- Beispiel SocketChannel

-
- Netzwerk-Programmierung erfordert die Definition von
 - geeigneten Leitungen
 - Übertragungsprotokollen
 - explizite Befehle zum Senden und Empfangen
 - kein Schutz von gemeinsamen Variablen erforderlich

Das Schema des Nachrichtenaustauschs ist in [Abbildung 2](#) dargestellt.

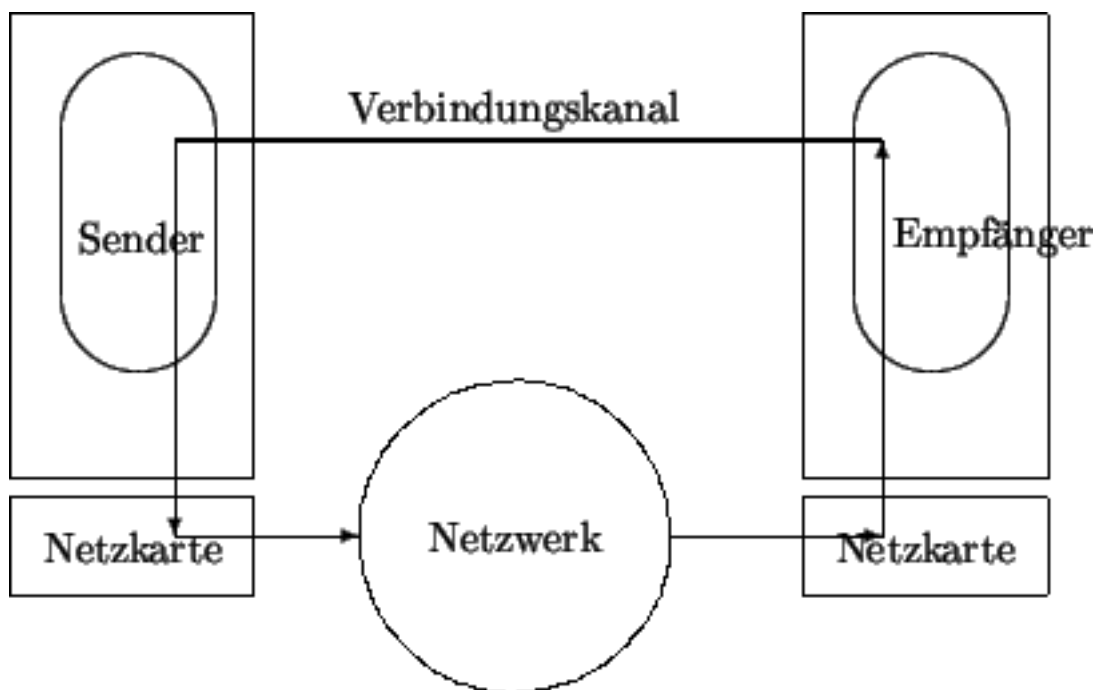


Abbildung 2: Schema des Nachrichtenaustauschs

- Java bietet Kommunikation auf Basis von TCP/IP Sockets
- von praktisch allen Betriebssystemen unterstützt
- nahezu jede Treiber-Software von Netzwerk Hardware bietet Unterstützung für TCP/IP
- Sockets sind aus Programmiersicht die Software-Schnittstelle zum Netzwerk
- entsprechen etwa den Filehandles
- Java-Netzwerk-Support im Package `java.net`
- zusammen mit dem Package `java.io`

- und dem Package java.nio
- bietet eine oder mehrere Punkt-zu-Punkt Verbindungen
- 1-zu-n- oder m-zu-n-Verbindungsnetzwerke nur mit zusätzlichem Aufwand bzw. Packages

5.1. Sockets

- Implementierung von Verbindungen mit
- Java Klassen ServerSocket und Socket
- Socket-Verbindung wird unsymmetrisch aufgebaut
- ein Ende (der Server Socket) wartet auf Verbindungswunsch
- anderes Ende (der Client Socket) versucht eine Verbindung aufzubauen
- wenn dies auf beiden Seiten gelingt besteht ein zuverlässiger Verbindungskanal
- ab dann Senden und Empfangen möglich

Spezifikation von ServerSocket:

```
public ServerSocket(int port) throws IOException
public Socket accept() throws IOException
```

- Konstruktor ServerSocket erzeugt einen neuen Server Socket an port
- bei Portnummer 0 an einem beliebigen freien Port
- Methode accept() wartet auf einen Verbindungswunsch
- gibt einen Socket für die Verbindung zurück
- blockiert bis eine Verbindung zustande kommt

Spezifikationen von Socket

```
public Socket(String host, int port)
    throws UnknownHostException, IOException
public InputStream getInputStream() throws IOException
public OutputStream getOutputStream() throws IOException
```

- Konstruktor Socket erzeugt neuen Client Socket
- zu dem angegebenen host und port
- stellt Eingabe- und Ausgabe-Strom zur Verfügung
- Zugriff mit getInputStream() und getOutputStream()
- Strom (Stream) ist eine unformatierte und unstrukturierte Folge von Daten
- an einem Ende werden Daten eingefüllt, am anderen Ende erscheinen sie in der gleichen Reihenfolge
- InputStream und OutputStream bestehen aus Folgen von Bytes

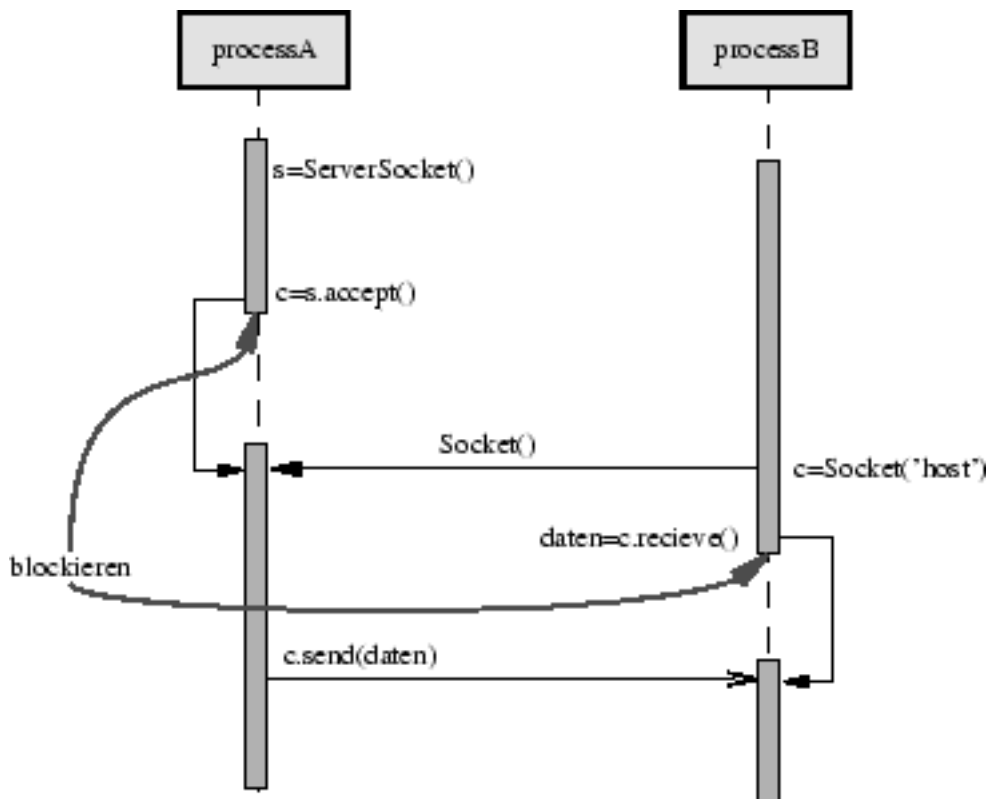


Abbildung E: UML Socket

Zuordnung von passenden Datenströmen:

- reine Unicode Zeichen mit Reader und Writer
- im folgenden ObjectOutputStream
- Austausch beliebiger serialisierbarer (serializable) Objekte
- Umwandlung von Java-Objekten in einen Byte-Strom
- und typsicheres Versenden und Empfangen
- auch für Datei-Ströme verwendbar
- akzeptiert nur Objekte, die das java.io.Serializable Interface implementieren
- ist eins der wesentlichen neuen Features von Java 1.1.
- automatische Serialisierung überschreibbar
- dann muß man die Klasse selbst kodieren und dekodieren
- bei Filehandles keine Serialisierung sinnvoll
- Strom-Header enthält eindeutige Identifikation der Objekt-Strom-Klasse

Spezifikation der benötigten Konstruktoren und Methoden

```

public ObjectOutputStream(OutputStream out) throws IOException
public void flush() throws IOException
public ObjectInputStream(InputStream in)
    throws IOException, StreamCorruptedException
    
```

- Konstruktor ObjectOutputStream erzeugt neuen Objekt-Ausgabestrom

- zu einem gegebenen OutputStream
- flush() verschickt Daten unmittelbar
- Konstruktor ObjectOutputStream erzeugt neuen Objekt-Eingabestrom
- zu einem gegebenen InputStream
- Passt die Identifikation nicht wird eine StreamCorruptedException ausgelöst
- z.B. inkompatible JDK-Version, inkompatible Serialisierung
- blockiert, bis ein Objekt-Ausgabe-Strom die entsprechenden Daten gesendet

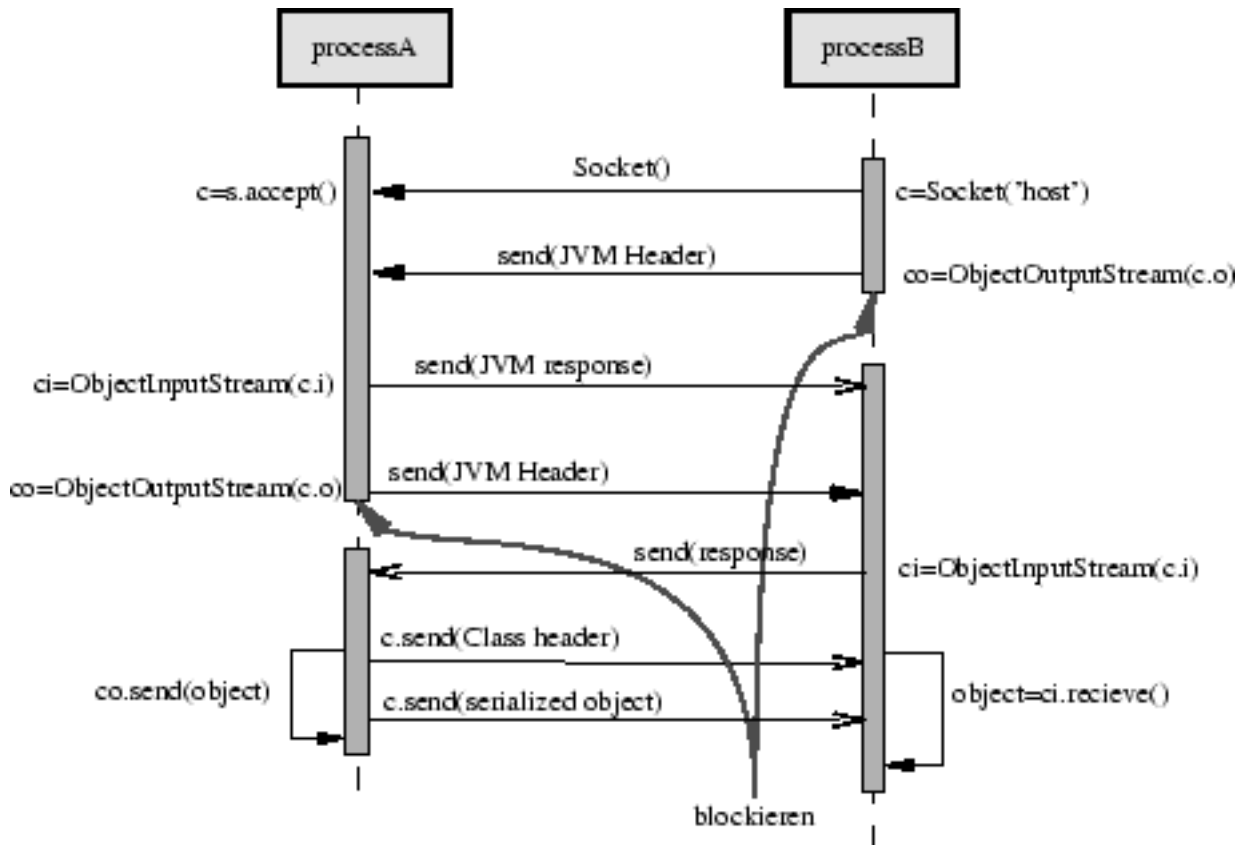


Abbildung F: UML Serialization

Datenübertragung mit Send-Operation (send) und Empfangs-Operation (receive).

Spezifikation aus ObjectOutputStream

```
public final void writeObject(Object obj)
    throws IOException
```

- writeObject() schreibt ein Objekt auf den Ausgabe-Strom
- der gesamte Graph des Objekts wird zerlegt (serialisiert)
- und mit dem Klassennamen und Klassensignatur geschrieben

Spezifikation aus ObjectInputStream

```
public final Object readObject()
    throws OptionalDataException,
           ClassNotFoundException, IOException
```

- readObject() liest ein Objekt von dem Eingabe-Strom
- der Klassenname und Klassensignatur wird gelesen
- der gesamte Graph des Objekts wird gelesen
- und rekonstruiert

5.2. SocketChannel

Der Konstruktor nimmt einen Socket als Eingabe und setzt die Objekt-Ströme aus den entsprechenden Strömen auf.

Die Methode close() dient zum schließen des Kanals.

send()- und receive()-Methoden dienen der Datenübertragung

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

public class SocketChannel {

    private ObjectInputStream in;
    private ObjectOutputStream out;

    private Socket soc;

    public SocketChannel(Socket s) throws IOException {
        soc = s;
        if (checkOrder(s)) {
            in = new ObjectInputStream(s.getInputStream());
            out = new ObjectOutputStream(
                s.getOutputStream());
        }
        else {
            out = new ObjectOutputStream(
                s.getOutputStream());
            in = new ObjectInputStream(s.getInputStream());
        }
    }

    public void close() {
        if (in != null) {
            try { in.close(); } catch (IOException e) { }
        }
        if (out != null) {
            try { out.close(); } catch (IOException e) { }
        }
        if (soc != null) {
            try { soc.close(); } catch (IOException e) { }
        }
    }

    private boolean checkOrder(Socket s)
        throws IOException {
        int p1 = s.getLocalPort();
        int p2 = s.getPort();
        if (p1 < p2) return true;
        else if (p1 > p2) return false;

        int a1 = s.getLocalAddress().hashCode();
        int a2 = s.getInetAddress().hashCode();
        if (a1 < a2) return true;
        else if (a1 > a2) return false;
    }
}
```

```
        throw new IOException(); // this shouldn't happen
    }
}
```

- die Reihenfolge der Erzeugung der Objekt-Ströme out und in muß vertauscht sein
- da sich Objekt-Ströme bei dem Verbindungsaufbau über die Verwendung der gleichen Klassen und JDK-Versionen einigen müssen
- Methode checkOrder vertauscht
- auf dem gleichen Host sind die Ports verschieden, sonst die IP-Adressen

Die Sende- und Empfangs-Methoden sind wie folgt.

```
public void send(Object v) throws IOException {
    out.writeObject(v);
}
```

Im Fehlerfall wird eine IOException ausgelöst, die dann vom Aufrufer behandelt werden muß.

```
public Object receive()
    throws IOException,
        ClassNotFoundException {
    return in.readObject();
}
```

Neben einer IOException kann auch noch eine ClassNotFoundException ausgelöst werden, falls das Objekt zu einer dem Empfänger unbekannten Klasse gehört.

Beispiel HelloWorld mit SocketChannel

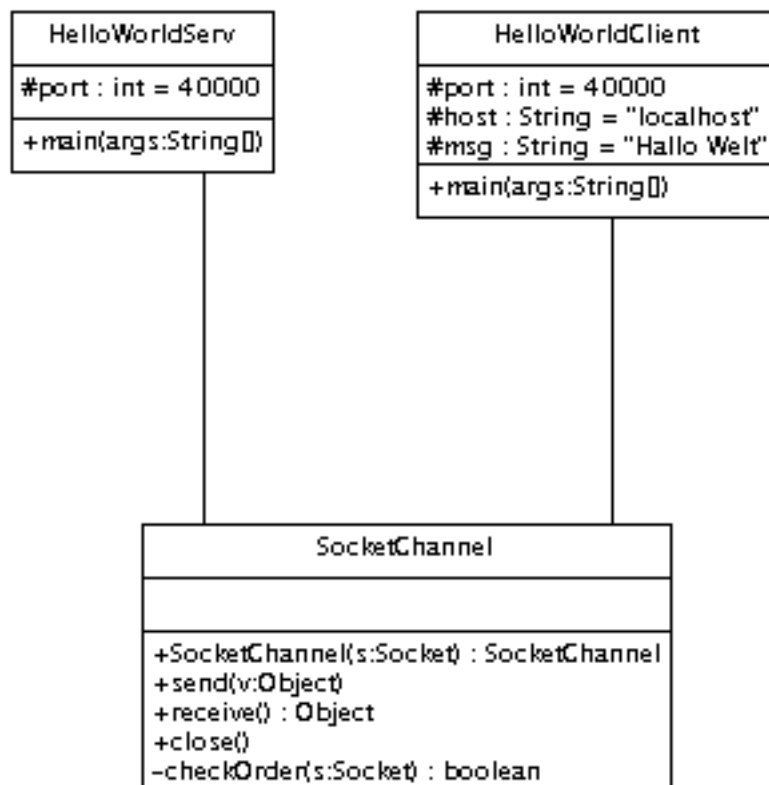


Abbildung A1: HelloWorld

Implementierung: [HelloWorldClient.java](#) [HelloWorldServ.java](#) [SocketChannel.java](#)

© Universität Mannheim, Rechenzentrum, 2000-2004.

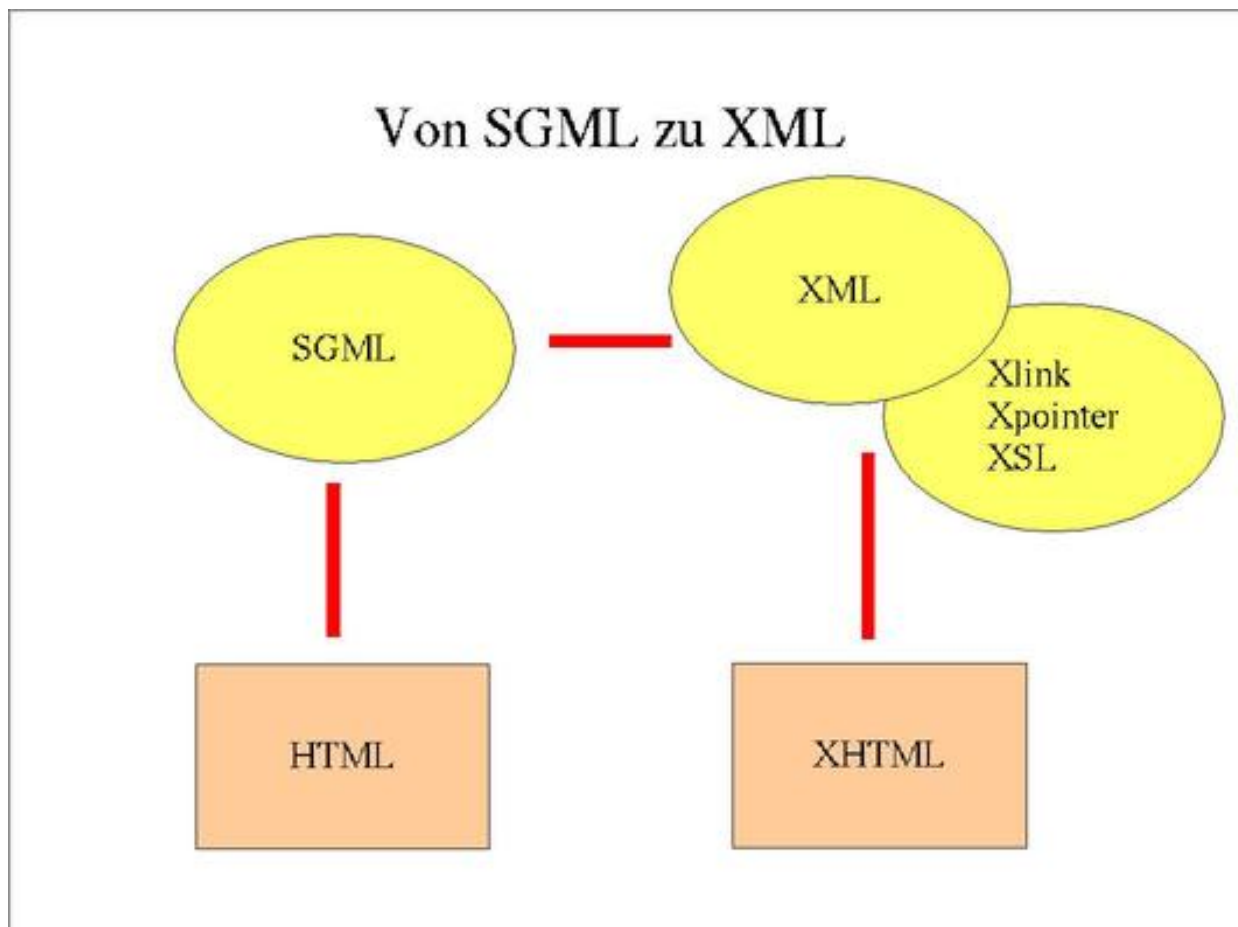
Last modified: Tue May 25 18:04:32 CEST 2004

6. Extensible Markup Language (XML)

- Einleitung
- XML Sprachkonstrukte
- XML Anwendungen
- Stand und Ausblick

6.1. Einleitung

- HTML
- SGML
- XML



- eXtensible Markup Language
- W3C Recommendation,

10. Feb. 1998

- von SGML, TEI, HTML Leuten
- Basis für viele Dinge

HTML Beschränkungen

- nicht erweiterbar
- nur einfache Struktur
- keine Validierung

HTML Beispiel (1)

```
<HTML>
<HEAD>
<TITLE>Beispiel
</TITLE>
</HEAD>
<BODY>
<H1>Überschrift</H1>
<P>Ein Paragraph
mit Text ...</P>
</BODY>
</HTML>
```

HTML Beispiel (2)

```
<UL>
<LI>Eine ungeordnete
<LI>Liste
<LI>it verschiedenen
<LI>Punkten
</UL>
```

- Eine ungeordnete
- Liste
- mit verschiedenen
- Punkten

Komplexität von SGML

- Metasprache zur Definition von HTML
- komplexe Struktur
- Definition von optionalen Tags
- strenge Validierung
- Spezifikation: 200+ Seiten
- XML Spezifikation: 40 Seiten

SGML Beispiel (1)

```
<!ELEMENT COMPANY - - (NAME?,PRODUCT?) >
<!ELEMENT NAME - 0 (#PCDATA)>
<!ELEMENT PRODUCT - - (ITEM*)>
<!ELEMENT ITEM - 0 (#PCDATA)>
<!ENTITY ... >
<!ATTLIST COMPANY
  type (non-profit | limited | corp) >
```

SGML Beispiel (2)

```
<COMPANY type=corp >
<NAME>All you want
<PRODUCT>
<ITEM>Apartments
<ITEM>Automobiles
<ITEM>...
</PRODUCT>
</COMPANY>
```

Ansatz von XML

- Teilmenge von SGML
- einfache Spezifikation
- offener Standard
- International, Unicode
- keine optionalen Tags
- Syntax Prüfbar
- Validierbar
- effizient
- verbesserte Links, XLL
- Style Sheets, XSL



XML im Beispiel

Dokument: .xml

```
<?xml version="1.0"?>
<!DOCTYPE personals SYSTEM "personal.dtd">
<personals>
  <person id="H.MARUYAMA" >
    <name><family>MARUYAMA</family> <given>Hiroshi</given></name>
    <email>maruyama@jp.ibm.com</email>
    <link subordinates=" N.URAMOTO K.TAMURA "/>
  </person>
  <person id="N.URAMOTO">
    <name><family>URAMOTO</family> <given>Naohiko</given></name>
    <email>uramoto@jp.ibm.com</email>
    <link manager=" H.MARUYAMA"/>
  </person>
  <person id="K.TAMURA">
    <name>
      <family>TAMURA</family> <given>Kent</given>
    </name>
```

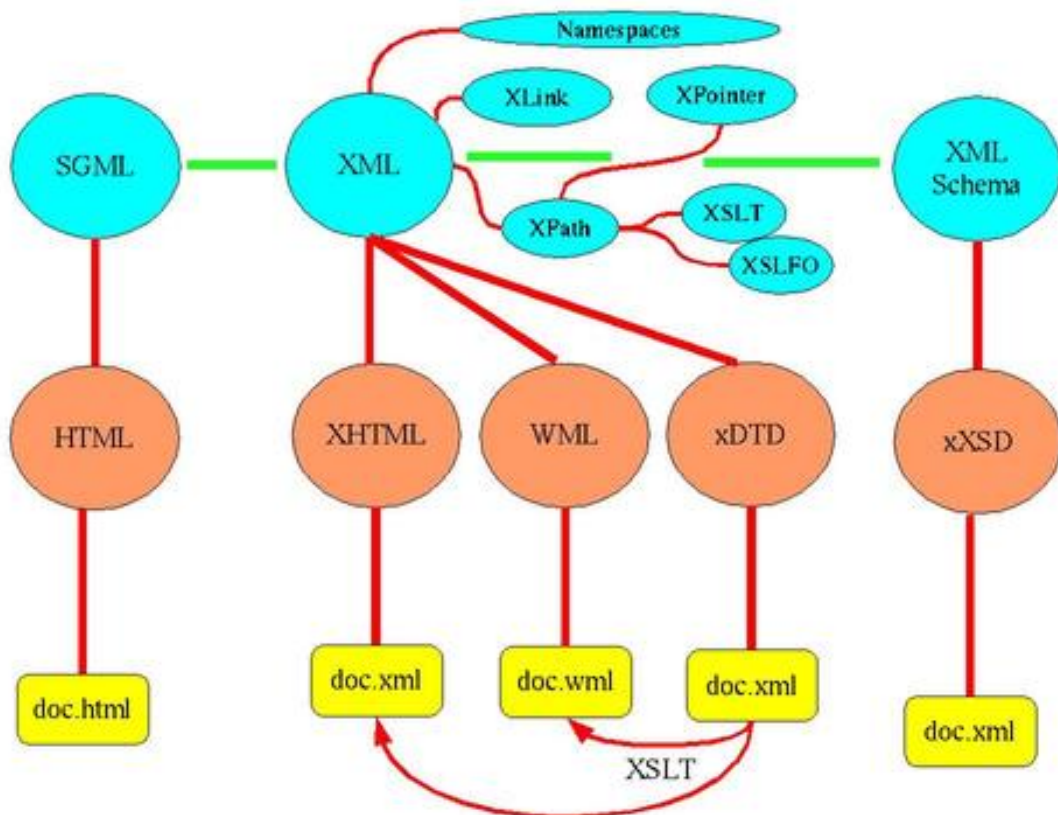
Extensible Markup Language (XML)

```
<!-- This URL is mail address.-->  
<url href="mailto:kent@trl.ibm.co.jp"/>  
<url href="mailto:tkent@jp.ibm.com"/>  
<link manager="H.MARUYAMA"/>  
</person>  
</personals>
```

Description / Definition: .dtd

```
<?xml encoding="US-ASCII"?>  
  
<!ELEMENT personals (person)+>  
<!ELEMENT person (name,email*,url*,link?)>  
<!ATTLIST person id ID #REQUIRED>  
<!ELEMENT family (#PCDATA)>  
<!ELEMENT given (#PCDATA)>  
<!ELEMENT name (#PCDATA|family|given)*>  
<!ELEMENT email (#PCDATA)>  
<!ELEMENT url EMPTY>  
<!ATTLIST url href CDATA #REQUIRED>  
<!ELEMENT link EMPTY>  
<!ATTLIST link  
  manager IDREF #IMPLIED  
  subordinates IDREFS #IMPLIED>
```

XML "Helfer"



Extensible Markup Language (XML)

- XML-Namespace, Namensräume



- XLink, XML Linking Language, Verweise aus XML Dokumenten heraus



- XPointer, XML Pointer Language, Zeiger in XML Dokumente hinein



- XPath, XML Path Language, Wird in XSLT und XPointer benötigt, Zeichenketten die einen bestimmten Teil eines XML Dokuments bezeichnen



- XSL, Extensible Stylesheet Language Stildefinitionen
- XSLT, XSL Transformations XML Stiltransformationen



- XML Schema: Structures, Datatypes Bedingungen (Constraints) für Dokumentstrukturen und Datentypen



6.2. XML Sprachkonstrukte

- Document Type Definition
- "wellformed" XML Dokumente
geht ohne DTD
- "valid" XML Dokumente
nur bezüglich einer DTD
- Referenz per URI oder Inline

Unterschiede zu HTML

- XML verlangt korrekte Schachtelung der Elemente
- leere Elemente sind speziell gekennzeichnet
- nur ein einziges "root" Element
- Attributwerte in Quotes
- Entities brauchen eine DTD
- Elementnamen sind Case-sensitiv
- White-Space im Inhalt ist relevant
- mehrere Zeichensätze sind verwendbar
- es gibt nur wenige reservierte Zeichen

XML Dokument

- XML Deklaration

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

- Deklaration des Dokumenttyps

```
<!DOCTYPE book SYSTEM fileurl >  
<!DOCTYPE memo [ ... ] >  
<!DOCTYPE html PUBLIC bezeichner url >
```

- Dokument-Instanz

```
<book>  
<head> ... </head>  
<body>  
...  
</body>  
</book>
```

Dokumentbeschreibung

- Elemente

```
<!ELEMENT name Inhalt >  
<name> ... </name>
```

- Attribute

Extensible Markup Language (XML)

```
<!ATTLIST element AttName AttTyp Default >
<element AttName="wert" >
```

- Entities

```
<!ENTITY name wert >
&name;
<!ENTITY % name wert >
%name;
```

- Character Data, CDATA

```
<![CDATA[ dies ist kein <tag/> ]]>
```

- Parsed Character Data, PCDATA
Mischung aus CDATA und Elementen
- Anweisungen, Processing Instructions

```
<?php ... ?>
```

- Kommentare

```
<!-- ... -->
```

- Leerraum, White-Space
ist im Inhalt signifikant

Inhaltsbeschreibung, Content Model Spec

- Folgen

```
(from, to, subject, message, signature)
```

- Auswahl

```
(para|list|image)
```

- Wiederholungsoperatoren

- + einmal oder mehr
- * nullmal oder mehr
- ? nullmal oder einmal

- beliebiger Inhalt

```
(#PCDATA)
```

- beliebige Zusammensetzungen

```
(head, (p|list|image)*, div+, (#PCDATA|em|strong))
```

- leeres Element

```
EMPTY
```

- beliebiges Element

```
ANY
```

Attributbeschreibung

- Attribut Typ
 - CDATA Zeichenketten
 - ID Bezeichner
 - IDREF Verweis auf Bezeichner
 - IDREFS Folge von IDREF
 - ENTITY
 - ENTITIES Folge von ENTITYs
 - NMTOKEN
 - NMTOKENS Folge von NMTOKENs
 - ENUMERATION Aufzählung von Werten
 - NOTATION Festlegung von Wertformaten
 - (EN|FR|GR) Enumeration, Aufzählung
- Attribut Default
 - #IMPLIED wird von der Anwendung erkannt
 - #REQUIRED muss angegeben werden
 - #FIXED wert fester Wert
 - wert Defaultwert
- Beispiel

```
jahr #FIXED      "2000"  
lang (EN|FR|GR) "EN"  
bez ID          #IMPLIED  
href CDATA      #REQUIRED
```

Beispiel

Beschreibung des Dokumententyps

```
<?xml encoding="US-ASCII"?>  
  
<!ELEMENT personals (person)+>  
<!ELEMENT person (name,email*,url*,link?)>  
<!--ATTLIST person id ID #REQUIRED-->  
<!ELEMENT family (#PCDATA)>  
<!ELEMENT given (#PCDATA)>  
<!--ELEMENT name (#PCDATA|family|given)*-->  
<!--ELEMENT email (#PCDATA)-->  
<!--ELEMENT url EMPTY-->  
<!--ATTLIST url href CDATA #REQUIRED-->  
<!--ELEMENT link EMPTY-->  
<!--ATTLIST link  
manager IDREF #IMPLIED  
subordinates IDREFS #IMPLIED-->
```

XML Dokument entsprechend dieser DTD

```
<?xml version="1.0"?>  
<!DOCTYPE personals SYSTEM "personal.dtd">
```

Extensible Markup Language (XML)

```
<personals>
...
<person id="K.TAMURA">
  <name>
    <family>TAMURA</family>
    <given>Kent</given>
  </name>
  <!-- This URL is mail address.-->
  <url href="mailto:kent@trl.ibm.co.jp"/>
  <url href="mailto:tkent@jp.ibm.com"/>
  <link manager="H.MARUYAMA"/>
</person>
</personals>
```

Validierung

Überprüfung ob der Inhalt eines XML Dokumentes einer gegebenen DTD entspricht.

Die Java Klassen zur Verarbeitung von XML-Daten gehören ab J2SDK 1.4 standardmässig zum Lieferumfang. Bis zum J2SDK 1.3 muss ein zusätzliches Softwarepaket installiert werden.

Zur Überprüfung benutzen wir Shell-Skripte.

JDK 1.4 und die xercesSamples.jar von xml.apache.org

valid (JDK 1.4 Version):

```
#!/bin/sh
VALIDPATH="/home/.../java/lib/xercesSamples.jar"
export CLASSPATH="$VALIDPATH:$CLASSPATH"
/usr/java/j2sdk1.4.1_01/bin/java sax.Counter -p org.apache.crimson.parser.XMLReaderImpl -v $*
```

valid.bat (JDK 1.4 Version):

```
set VALIDPATH=u:\xerces\xercesSamples.jar
set CLASSPATH=%VALIDPATH%;%CLASSPATH%
java sax.Counter -p org.apache.crimson.parser.XMLReaderImpl -v %1
```

Hilfe:

```
usage: java sax.Counter (options) uri ...

options:
-p name      Select parser by name.
-x number    Select number of repetitions.
-n | -N      Turn on/off namespace processing.
-np | -NP    Turn on/off namespace prefixes.
              NOTE: Requires use of -n.
-v | -V      Turn on/off validation.
-s | -S      Turn on/off Schema validation support.
              NOTE: Not supported by all parsers.
-f | -F      Turn on/off Schema full checking.
              NOTE: Requires use of -s and not supported by all parsers.
-dv | -DV    Turn on/off dynamic validation.
              NOTE: Requires use of -v and not supported by all parsers.
-m | -M      Turn on/off memory usage report
-t | -T      Turn on/off "tagginess" report.
--rem text   Output user defined comment before next parse.
-h           This help screen.
```

Verwendung:

```
valid datei.xhtml
```

Alternativ können Sie die Klasse `Counter.java` selbst kompilieren und wie gewohnt benutzen:

```
java Counter -v datei.xhtml
```

6.3. XML Anwendungen

in Wissenschaft und EDV

- **SVG** Scalable Vector Graphics, Vektorgrafik
- **SMIL** Synchronized Multimedia Integration Language
- **CML** Chemical Markup Language
- **MML** Mathematical Markup Language
- **RDF** Resource Description Framework
PICS, CDF, CRP
- **OSD** Open Software Distribution

im E-Commerce und Finanzwirtschaft

- **OASIS**, Organization for the Advancement of Structured Information Standards
- **OFX**, Open Financial Exchange
- **CommerceNet**, EDI B2B, e-Commerce Framework, EDI via XML
- **Ariba - cXML**, Transaktionen: Aufträge, Rechnungen, Änderungsaufträge
- **finXML**, XML für Finanzmärkte, Zinssätze, Währungsumtausch, Bonds, Geldmärkte, Anlagen, Optionen
- **Acord**, XML Standards für die Versicherungswirtschaft
- **Rosettanet**, IT supply chain alignment, server-to-server business exchange
- **OBI**, Open Buying Initiative
- Open Travel Alliance
- Open Trading Protocol
- Open Financial Exchange

6.4. Stand und Ausblick

XML Tools

- **SAX**
Simple API for XML
- **XML4J**
XML-Parser in Java von IBM, mit DTD und DOM1, ist in Xerces aufgegangen
- **msxml**
XML-Parser in Java von Microsoft, mit DOM1 ohne DTD
- **Xalan, Xerces**
Tools der Apache XML Aktivitäten
- **Koala**
XSL Processor, wird nicht mehr weiter entwickelt

- **Lotus**
XSL Processor, ist in Xalan aufgegangen

Stand der Entwicklung

26. April 2004

- XML 1.0,
W3C Recommendation, 10. Feb. 1998
Second Edition, 6. October 2000
- XML 1.1,
W3C Recommendation, 4. Feb. 2004
Last edited 15. April 2004
Änderungen bezüglich der Zeichensätze Unicode, UTF-8, UTF-16
- XSL Transformations (XSLT) 1.0,
W3C Recommendation, 16. Nov. 1999
- XML Path Language (XPath) 1.0,
W3C Recommendation, 16. Nov. 1999
- XML Names, Namespaces in XML
W3C Recommendation, 17. Jan. 1999
- Associating Style Sheets with XML documents
W3C Recommendation, 29. June 1999
- XSL 1.0, (jetzt nur Formatting)
W3C Recommendation, 15. Oktober 2001
- XML Linking Language (XLink),
W3C Recommendation, 2. Juni 2001
- XML Base (XBase),
W3C Recommendation, 27. June 2001
- XML Pointer Language (XPointer),
W3C Recommendation, 25 March 2003
- XML Schema, Part 1 Structures, Part 2 Datatypes,
W3C Recommendation, 2. Mai 2001
- XML Protocol, SOAP
SOAP 1.1, W3C Note, 8. May 2001
SOAP 1.2, W3C Recommendation, 24. Juni 2003
- XML Query 1.0 and XPath 2.0 Data Model,
W3C Working Draft, 12. November 2003
- XML Path Language (XPath) 2.0,
W3C Working Draft, 12. November 2003
- XML Signature Syntax and Processing,
W3C Recommendation, 12. February 2002,
IETF Request for Comments: 3275, March 2002
- Canonical XML 1.0,
W3C Recommendation, 15 March 2001
- XML Information Set (Second Edition),
W3C Recommendation, 4. February 2004
- XML Forms 1.0,
W3C Recommendation, 14. Oktober 2003
- Web Services Description Language (WSDL) 1.2,
W3C Working Draft, 11. Juni 2003
- Web Ontology Language (OWL) 1.0,
W3C Recommendation, 10. Februar 2004
- RDF Schema,

W3C Recommendation, 10. Februar 2004

- RDF Site Summary (RSS) 1.0,
(RSS WG) Recommendation, 9. Dezember 2000

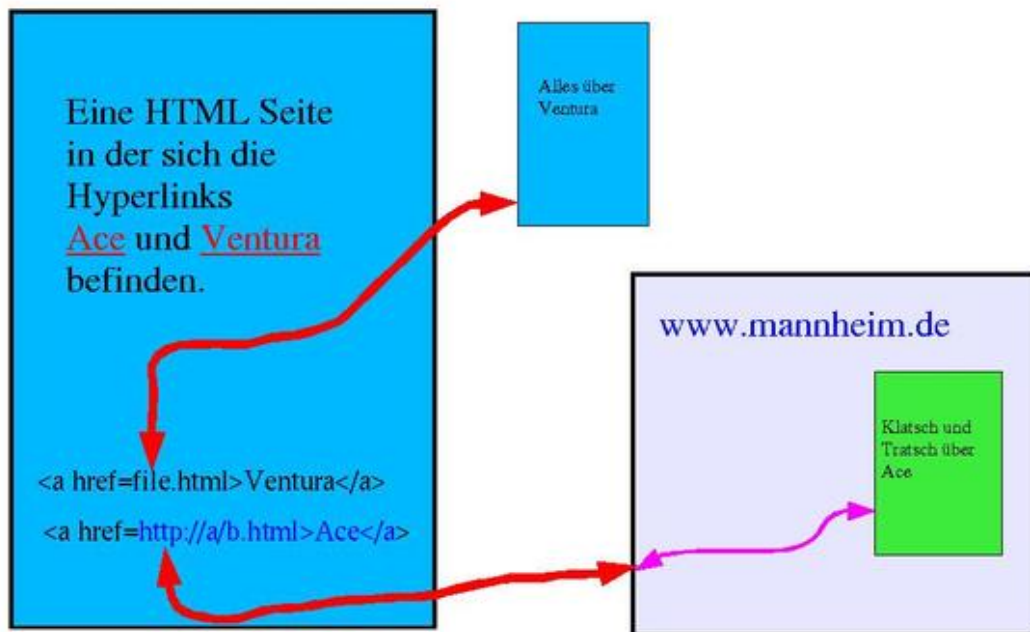
© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat Jun 12 11:52:23 CEST 2004

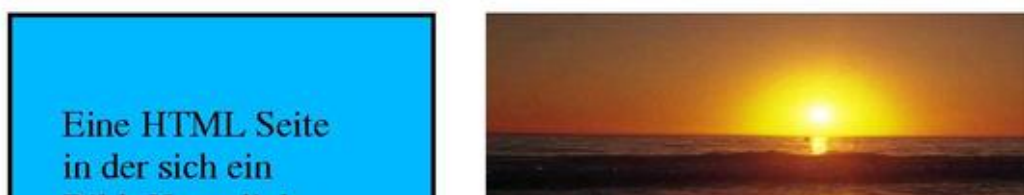
7. Hypertext Markup Language (HTML)

- Einordnung
- Elementares HTML
- Volles HTML
- Ausblick

`<a href=URL>Beschreibung</a>`



`<img src=URL alt=Bildbeschreibung>`



7.1. Einordnung

- Standard Generalized Markup Language (SGML)
- Extended Markup Language (XML)
- WYSIWYG im Gegensatz zu Markup
- Orientierung am Bildschirm
- Problem: Drucken

Entwicklung

- erster Ansatz von Tim Berners-Lee, 1989
- erster Browser, und Server 1990/91
- HTML 1.0, erster Entwurf 1992
- Web-Technologie Public Domain, 1993
- HTML 2.0, 1995, November
- HTML 3.2, 1997 Januar
- HTML 4.0, 1997 Dezember
- HTML 4.01, 1999 Dezember
- XHTML 1.0, 2000 Januar
- XHTML Basic, 2000 Dezember
- XHTML 1.1 - Module based XHTML, 2001 Mai
- XHTML 2.0, Working Draft, 2003 Januar



Web-Zutaten

1. Identifikation von Objekten: URL, URI
2. Übertragung von Objekten: Protokolle, z.B. HTTP
3. Hypertext: Darstellung und Navigation, z.B. HTML

HTML-Konzeption

1. Dokumentstruktur durch Markup
2. Dokument-Darstellung, Präsentation durch Browser
3. Interaktion

7.2. Elementares HTML

HTML 4.0 ist eine SGML Anwendung

Festlegung in der Document Type Definition (DTD)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd" >
```

Syntax

- Zeichensatz: ISO/IEC 10646, Universal Character Set (UCS), Unicode 2.0
- Elemente: <name> Inhalt </name>
- Markup oder Textauszeichnungen: *Tags* werden in spitze Klammern eingeschlossen: <tag>
- bei einigen Elementen ist das Ende-Tag optional manchmal auch das Start-Tag, z.B. BODY
- einige Elemente sind ohne Inhalt, d.h. leer
- für XML auch kombinierte Beginn-Ende-Tags: <tag/>
- Attribute: attname="wert"
<tag att1="wert1" att2="wert2" >
- Groß-Klein-Schreibung wird ignoriert
- Kommentare: <!-- ignoriierter Text -->
- spezielle Zeichen: < (<) > (>) ¨ (ä) & (&) å (å)

Dokumentstruktur

- Grammatik von HTML
- Schachtelung der Elemente
- Festlegung in der Content-Model Spezifikation
- Beispiel in EBNF:
HTML = HEAD BODY
UL = (LI)+
- in SGML:
<!ELEMENT HTML O O (HEAD BODY) >
<!ELEMENT UL - - (LI)+ >
<!ELEMENT IMG - O EMPTY >
- in XML:
<!ELEMENT html (head body) >
<!ELEMENT ul (li)+ >
<!ELEMENT img EMPTY >

Darstellung

- durch Browser
- Mosaic (April 1993), Netscape, MS Internet Explorer, W3C Amaya, Lynx, html2ps, Mobil-Telefone, Palm-Tops, Sprachausgabe, Blindenschrift
- unbekannte Elemente werden ignoriert
nur der Inhaltsteil wird dargestellt
- unbekannte Attribute werden ignoriert
- bei unbekannten Attribut-Werten wird der Default genommen

Uniform Resource Locator (URL)

- dienst://hostname/pfad/object.ext

Hypertext Markup Language (HTML)

- Dienst-Selektor
- Hostname
- Pfad-Selektor
- Objekt im Dokument: doc.html#part_2
- Relative URIs: ../images/bild.gif
oder adoc.html

Elemente

Element	Bedeutung	Content-Model
html	SGML Anwendung	head body
head	Kopfinformationen	title (script style meta link)+
body	eigentlicher HTML Teil	alles ohne Kopfelemente
title	Titel, Inhaltsbezeichnung	Text
h1, ..., h6	Überschriften	Text
a	Verweis, Link, Anker	Linktext etc.
p	Paragraph	alles ohne Block-Elemente
ul, ol	Listen: unnummeriert, nummeriert	li+
dl	Definitionslisten	(dt dd)+
li, dt, dd	Listenelement, Definitionsterm, Definition	alles mit Ausnahmen
img	Bild im Text	empty
pre	vorformatierter Text	Text
br	Zeilenumbruch	EMPTY
em, strong, code	Betonung, Hervorhebung, Auszeichnung	Text
blockquote, q	Einrückung	Text

unvollständige Auswahl

Blanks und Zeilenumbrüche im Dokument werden (mit Ausnahmen) ignoriert

Attribute

Attribut	Bedeutung	Werte
href	Link URI in a	Text
name	Ziel von einem Link in a	Text
src	Bild URI in img	Text
alt	Alternative Bezeichnung in img	Text
align	Textausrichtung in p	(left center right)
value	Nummer in ol li	Zahl
type	Nummerstil in ul li	(disk square circle)
title	zusätzliche Beschreibung	Text
bgcolor	Hintergrundfarbe	black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive,

		yellow, navy, blue, teal, aqua oder #rrggbb
--	--	--

Beispiele

7.3. Volles HTML

Erweiterungen

- Formulare
- Tabellen
- Frames
- Style Sheets, Darstellungsvorlagen
- Scripte, Eingebettete Programme
- Eingebettete Objekte, Applets
- Internationalization, I18N
Unicode, Textrichtung

Elemente

Element	Bedeutung	Content-Model
form	Formulare	(input select other)+
input	Eingabefeld im Formular	EMPTY
select	Auswahl-Menue	(optgroup option)+
option	Auswahl-Option	Text der Option
textarea	Eingabebereich im Formular	Anfangstext
table	2-D Tabellen	caption, tr+
tr	Tabellen Zeile	(th td)+
th, td	Tabellen Element	alles mit Ausnahmen
map	Bilder mit Links verknüpfen	area+ other
area	Beschreibung der Bildkoordinaten	EMPTY
html	SGML Anwendung	head frameset
frameset	Aufteilung der Anzeigefläche	frame+ noframes
frame	separate Anzeigefläche	EMPTY
noframes	alternative ohne Frames	alles
link	Verweise, mit Rollenangaben, z.B. auf Style Sheet	EMPTY
style	Stilangabe	Stil-Script
script	JavaScript Programm im Text	Programm

Hypertext Markup Language (HTML)

applet	Java Programm wie Bild	param+
param	Parameter für Java Programm	EMPTY
object	Programm oder Bild	param+
meta	Information über die Seite	EMPTY
div	Dokument-Abschnitt, Block	alles
span	Text-Abschnitt, Inline	alles ohne Block-El.
del, ins	Text Revisionen	alles mit Ausnahmen

Attribute

Attribut	Bedeutung	Werte
action	URL eines (CGI-)Programms in form	Zeichenkette
method	HTTP Methode in form	(get post)
type	Typ eines Eingabefelds in input	(text password radio checkbox submit reset file hidden image button)
selected, value	Option mit Wert in option	selected, Wert
border	Rahmentyp	(solid none dashed n)
height, width	Grösse	Pixel, Prozent, Meter
cellpadding, cellspacing	Abstand zwischen Text und Rand Abstand zwischen Zellen	Pixel, Prozent, Meter
usemap	Verweis auf zugeordnete map in img	Bezeichner
shape	Typ der area	circle rect poly
coords	Koordinaten der area	Liste
rows, cols	Spezifikation der Zeilen, Spalten in frameset	Pixel, Prozent, Meter
name	Name in frame	Bezeichner
frameborder	Rahmentyp eines Frames	(solid none dashed n)
target	Zielframe zur Anzeige der Seite in a	Bezeichner
rel, rev	z.B. Stilverweis in link	stylesheet next prev
style	Stilinformation fast überall	CSS Text
class	CSS-Klassen-Identifikator, fast überall	Bezeichner
id	Identifikator wie name, fast überall	Bezeichner
language	Programmiersprache in script	JavaScript VBScript other
code, classid	auszuführender Programmcode in applet, object	Bezeichner

codetype	Typ des Programmcodes in applet, object	Bezeichner
name	Metainfo. Bezeichnung, in meta	Text
http-equiv	Metainfo. HTTProtokol, in meta	Text
content	Metainfo. Inhalt, in meta	Text
lang	Sprache in einem Element	z.B. en, de
dir	Textschreibrichtung	z.B. ltr, rtl

Script-Ereignisse

Ereignis	Bedeutung
onload, onunload	Aufruf beim Laden
onchange	Aufruf bei Änderungen
onclick	Aufruf beim Anklicken
onmouseover	Maus über dem Element
onmouseout	Maus verläßt Element
onselect	bei Auswahl
onsubmit	beim Versenden

Beispiele

7.4. Allgemeines zum Web-Design

- Web-Seiten bieten neue Möglichkeiten aber auch Unterschiede z.B. zur Buchproduktion.
- Welche Zielgruppe?
- auf jeder Seite: Namen, Email, evtl. Copyright, Datum.
- Konzept für dauerhafte URLs.
- klares, durchsichtiges Hypertext-Konzept, Navigation.
- prüfen Sie alle Links.
- Navigationshilfen: Home, Site-Map, Stichworte, Suchmöglichkeiten.
- gute, einheitliche Gestaltung, klare Farbkonzepte, Farbkontraste (==> CSS).
- benutzen Sie Stilvorlagen (CSS).
- achte auf Ladezeiten, 30-40KByte ok.
- vermeide zu grosse Graphiken, insbesondere auf der Start-Seite.
- verwende explizite Grössenangaben bei Tabellen und Graphiken (width, height).
- verwende Thumbnails (daumengrosse Mini-Graphiken) mit Link auf die "richtige" Graphik.
- beachte auch nicht-visuelle Browser: Lynx, Organizer (alt, summary).
- beachte geringere Bildschirmauflösung (640x480, 800x600).
- Testen Sie mit unterschiedlichen Browsern.
- Testen Sie mit unterschiedlichen Bildschirmauflösungen.
- Lassen Sie Ihre Seiten testen.
- Halten Sie die Seiten aktuell.

- Denken Sie ans Drucken.

7.5. Ausblick

- Extensible HTML (XHTML)
- HTML-Editoren und Tools
- Cascading Style Sheets (CSS)
- JavaScript
- Document Object Model (DOM)

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Wed May 26 23:42:49 CEST 2004

8. Extensible Hypertext Markup Language (XHTML)

- HTML als XML Anwendung
 - HTML Tidy
 - XHTML Basic 1.0
 - XHTML 1.1
-

8.1. HTML als XML Anwendung

- HTML (3.2, 4.0) ist eine SGML Anwendung
- XML ist der Standard für erweiterbares Markup
- HTML muss in XML reformuliert werden
- Version 1.0, W3C Proposed Recommendation August 1999
- HTML 4.01 enthält die notwendigen Anpassungen
- im Januar 2000 als W3C Recommendation verabschiedet



Warum XHTML?

- XHTML Dokumente sind XML konform. Sie können mit XML Tools bearbeitet werden.
- XHTML Dokumente können als `text/html` von HTML 4.0 Browsern verwendet werden.
- XHTML Dokumente können aber auch als `text/xml` oder als `application/xml` (mit geeigneten Style Sheets) verwendet werden.
- XHTML Dokumente können mit DOM bzw. XML-DOM verwendet werden, d.h. mit (Java)Scripts und Applets.
- XHTML Dokumente verschiedener Autoren (Systeme, Umgebungen) werden besser zusammenpassen als HTML Dokumente.
- Da XHTML eine XML Anwendung ist, können *neue Markup-Elemente* einfach hinzugefügt werden.
- XHTML ist nicht mehr nur auf Browser beschränkt. Viele andere User-Agents (Handys, Sprachausgabe, etc.) werden damit umgehen können (best effort content transformation).

Bedingungen für XHTML konforme Dokumente

- Sie müssen entsprechend einer XHTML DTD gültige (valid) XML Dokumente sein.
- Das Root-Element muss `<html>` sein.
- Das Root-Element muss einen gültigen XHTML Namensraum bestimmen, der ein gültiger XML Namensraum sein muss.
- Es muss eine XML DOCTYPE Deklaration vor dem Root-Element vorhanden sein.
- Die Internet Medien Typen (Mime Types) dürfen `text/html`, `text/xml` oder `application/xml` sein.

Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
<head>
<title>Browser Titel</title>
</head>
<body>
<h1>Dokument Titel</h1>
<p>
Ein Paragraph <br />
auf zwei Zeilen.
</p>
<math xmlns="http://www.w3.org/TR/REC-MathML">
... Text in MathML ...
</math>
</body>
</html>
```

Verhalten von XML User Agents (Browser)

- XML-UAs müssen *well-formedness* feststellen.
- Validierende UAs müssen das Dokument gegen alle DTDs *validieren*.
- Bei unbekannten Elementen muss der Inhalt dargestellt werden (wie bei HTML, nicht bei XML).
- Unbekannte Attribute müssen ignoriert werden (wie bei HTML, nicht bei XML).
- Bei unbekannten Attribut-Werten muss der Defaultwert verwendet werden (wie bei HTML, nicht bei XML).
- Unbekannte Entities müssen als Zeichenkette (&xyz;) dargestellt werden (wie bei HTML, nicht bei XML).
- Unbekannte Zeichen (Characters) müssen so dargestellt werden, dass klar ist das sie nicht bekannt, aber erkennbar sind (nicht bei HTML, nicht bei XML).
- Whitespace direkt nach einem Start-Tag und unmittelbar vor einem End-Tag muss ignoriert werden (falls nicht per XML etwas anderes bestimmt wurde).

Unterschiede zu HTML 4.0

- XHTML Dokumente müssen *well-formed* sein, d.h. gültige Schachtelungsstruktur haben.

```
<p>Paragraph <em>Hervorhebung</em></p>
statt
<p>Paragraph <em>Hervorhebung</p></em>
```

- Element- und Attribut-Namen müssen in Kleinbuchstaben geschrieben sein.

```
<li> statt <LI>
```

- End-Tags müssen immer vorhanden sein (falls nicht per XML das Element als EMPTY deklariert wurde).

```
<p>Paragraph</p> <p>weiterer Paragraph</p>
statt
<p>Paragraph <p>weiterer Paragraph
```

- Bei leeren Elementen ohne End-Tag muss das Start-Tag mit `</>` beendet werden.

```
<br />
```

- Attributwerte müssen in Anführungszeichen eingeschlossen werden. Auch bei Zahlenwerten.

```
<img ... width="300" />
statt
<img ... width=300 />
```

- Attributwerte müssen immer angegeben werden.

```
<dl compact="compact" >
statt
<dl compact >
```

- In Attributwerten wird Whitespace auf jeweils ein Blank verkürzt, bzw. am Beginn und Ende von Zeichenketten abgeschnitten.

```
alt="  Beschreibung      eines  Bildes  "
wird zu
alt="Beschreibung eines Bildes"
```

- Script-Texte müssen als CDATA markiert werden, falls sie < oder & enthalten.

```
<script>
  <![CDATA[
    ... Inhalt des Scripts
  ]]>
</script>
```

- SGML Ausschluss-Definitionen sind nur informell festgelegt.
z.B. das a-Element darf kein weiteres a-Element enthalten.
- Das name Attribut von HTML muss als XML id Attribut angegeben werden.

```
<a name="section1" id="section1" ... >
```

Diese Datei in [XHTML](#) und als [XML](#).

Tips und Hinweise

- Processing Instructions und Zeichensätze werden nicht von allen UAs erkannt
<?...>, UTF-8, UTF-16.
- Benutze Blanks vor />
Benutze
 statt
</br>
Benutze <p></p> statt <p />.
- Benutze externe Scripts, falls "<", "&" oder "]"> vorkommen.
- Verwende keine Zeilenumbrüche und mehrfache Leerzeichen in Attribut werten.
- Benutze lang und xml:lang gleichzeitig als Attribute.
- Benutze name="xyz" und id="xyz" gleichzeitig als Attribute für die Bezeichnung von Elementen.
- Benutze <?xml ... encoding="iso-8859-1"> und <meta http-equiv="Content-type" ... charset="iso-8859-1"> gleichzeitig für die Bezeichnung von Zeichensätzen.
- Einige UAs haben Probleme mit Booleschen Attributen.
- Problem bei DOMs:
HTML 4.0 DOM benutzt Grossbuchstaben,
XHTML 1.0 DOM benutzt Kleinbuchstaben,
XML 1.0 DOM benutzt Gross-/Klein-Buchstaben.
- Problem mit "&" in Attributwerten, z.B.
href=".../script.pl?n1=w1&n2=w2"
- Probleme mit Style Sheets (CSS):
Gross-/Klein-Schreibung von Elementen.

Ausblick

- Modularisierung von XHTML, d.h. zuschneiden auf bestimmte UAs

- Formalisieren der Bildung von Teilmengen und Erweiterungen.
- Dokument Profile
- Stand im Mai 2003
- XHTML Basic, 2000 Dezember
- XHTML 1.1 - Module based XHTML, 2001 Mai
- XHTML 2.0, Working Draft, 2003 Januar

8.2. HTML Tidy

- Tool zur Fehlersuche in HTML
- bietet auch Fehlerkorrektur
- kann Müll von HTML-Editoren entfernen
- kann HTML nach XHTML konvertieren
- offizielles Tool des W3C
- erkennt, prüft und korrigiert Dokumenttyp
- für fast alle Plattformen verfügbar
- Unterstützung von XML, ASP, PHP



Beispiele für die Arbeitsweise von HTML Tidy

Beispiel für schlechtes HTML [bad.html](#) und das Ergebnis nach Bearbeitung mit HTML Tidy [good.html](#).

Fehlermeldungen aus dem Beispiel

```
> tidy exam/bad.html >exam/good.html

Tidy (vers 19th October 1999) Parsing "exam/bad.html"
line 3 column 1 - Warning: inserting missing 'title' element
line 5 column 2 - Warning: replacing unexpected <h2> by </h1>
line 5 column 37 - Warning: discarding unexpected </h3>
line 7 column 42 - Warning: replacing unexpected </i> by </b>
line 8 column 15 - Warning: replacing unexpected </b> by </i>
line 10 column 45 - Warning: missing </i> before </h2>
line 12 column 4 - Warning: inserting implicit <i>
line 14 column 2 - Warning: missing </i> before <p>
line 14 column 4 - Warning: inserting implicit <i>
line 14 column 43 - Warning: discarding unexpected <a>
line 16 column 2 - Warning: missing </a> before <li>
line 16 column 2 - Warning: missing </i> before <li>
line 16 column 2 - Warning: inserting implicit <ul>
line 24 column 1 - Warning: unknown attribute "tidy"
line 30 column 1 - Warning: <img> lacks "alt" attribute

"exam/bad.html" appears to be HTML proprietary
15 warnings/errors were found!
```

The alt attribute should be used to give a short description of an image; longer descriptions should be given with the longdesc attribute which takes a URL linked to the description. These measures are needed for people using non-graphical browsers.

For further advice on how to make your pages accessible see "<http://www.w3.org/WAI/GL>". You may also want to try "<http://www.cast.org/bobby/>" which is a free Web-based service for checking URLs for accessibility.

HTML & CSS specifications are available from <http://www.w3.org/>
To learn more about Tidy see <http://www.w3.org/People/Raggett/tidy/>
Please send bug reports to Dave Raggett care of [<html-tidy@w3.org>](mailto:html-tidy@w3.org)
Lobby your company to join W3C, see <http://www.w3.org/Consortium>

Aufruf und Verwendung von HTML Tidy

```
> tidy [[options] files]*

tidy: file1 file2 ...
Utility to clean up & pretty print html files
see http://www.w3.org/People/Raggett/tidy/

options for tidy released on 19th October 1999
-config <file>    set options from config file
-indent or -i     indent element content
-omit or -o       omit optional endtags
-wrap 72          wrap text at column 72 (default is 68)
-upper or -u      force tags to upper case (default is lower)
-clean or -c      replace font, nobr & center tags by CSS
-raw             leave chars > 128 unchanged upon output
-ascii           use ASCII for output, Latin-1 for input
-latin1          use Latin-1 for both input and output
-iso2022         use ISO2022 for both input and output
-utf8            use UTF-8 for both input and output
-mac            use the Apple MacRoman character set
-numeric or -n   output numeric rather than named entities
-modify or -m    to modify original files
-errors or -e    only show errors
-quiet or -q     suppress nonessential output
-f <file>       write errors to <file>
-xml            use this when input is wellformed xml
-asxml          to convert html to wellformed xml
-slides         to burst into slides on h2 elements
-help or -h     list command line options
Input/Output default to stdin/stdout respectively
Single letter options apart from -f may be combined
as in: tidy -f errs.txt -imu foo.html
For further info on HTML see http://www.w3.org/MarkUp
```

Einige wichtige Optionen

markup: yes, no

Erzeugen des verbesserten Markups.

wrap: number

Zeilenumbbruch bei angegebener Spalte. 0 = abgeschaltet.

input-xml: yes, no

Einlesen als XML.

output-xml: yes, no

Ausgabe von XML.

output-xhtml: yes, no

Ausgabe von XHTML.

doctype: omit, auto, strict, loose or <fp>

Festlegen des DOCTYPE in der Ausgabe.

char-encoding: *raw, ascii, latin1, utf8 or iso2022*

Festlegen des Zeichensatzes in der Ausgabe.

fix-backslash: *yes, no*

Wandelt "\" in URLs zu "/".

word-2000: *yes, no*

Versucht Müll, der von Word 2000 produziert wird zu entfernen.

clean: *yes, no*

Versucht überflüssigen Präsentations-Markup durch Stilregeln (CSS) oder Struktur-Markup zu ersetzen.

logical-emphasis: *yes, no*

Ersetzt i durch em, b durch strong, impliziert clean.

enclose-text: *yes, no*

Fasst Text auf Body-Level in Paragraphen. Wichtig für funktionierende Stilvorlagen.

split: *yes, no*

Teilt die Datei an h2 Elementen in einzelne "Folien".

new-empty-tags: *tag1, tag2, tag3*

new-inline-tags: *tag1, tag2, tag3*

new-blocklevel-tags: *tag1, tag2, tag3*

new-pre-tags: *tag1, tag2, tag3*

Definition von neuen Tags der entsprechenden Art.

Beispiel für ein Config-File

```
/* HTML Tidy configuration file */
markup: yes
wrap: 0
doctype: strict
break-before-br: yes
logical-emphasis: yes
enclose-text: yes
/* eof */
```

Was in Arbeit ist

- Validierung aller Attribute
- Verbesserter XML Support
- Verbesserung der Zeichensatz Unterstützung
- Verbesserung der ASP und PHP Unterstützung
- Verbesserte Folien Erzeugung

8.3. XHTML Basic 1.0

Reduktion von XHTML 1.0 auf die Elemente und Attribute, die auch auf kleinen Geräten angezeigt werden können. Zum Beispiel

- Handys
- Fernseher
- PDAs
- Verkaufsautomaten
- Pager

- Fahrzeug Navigationssysteme
- Spielekonsolen
- Lesegeräte für digitale Bücher
- Uhren mit CPUs

Die gemeinsame Fähigkeiten dieser einfachen UAs ermöglichen folgende XHTML Elemente.

- einfacher Text (mit Überschriften, Paragraphen und Listen)
- Hyperlinks (a und link)
- einfache Formulare
- einfache Tabellen
- Bilder
- Meta-Information

Elemente und Attribute, die nur auf aktuellen grafischen (PC-) Systemen funktionieren sind weggelassen.

- keine Stilvorlagen (CSS)
- keine Scripte (JavaScript) und zugehörige Attribute
- nur einfache Fonts (Schreibmaschinenschriften)
- kein File-Upload und Bilder in Formularen
- keine geschachtelten Tabellen
- keine Frames

Der Document Type ist

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
```

Definierte Module

Structure Module*

body, head, html, title

Text Module*

abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var

Hypertext Module*

a

List Module*

dl, dt, dd, ol, ul, li

Basic Forms Module

form, input, label, select, option, textarea

Basic Tables Module

caption, table, td, th, tr

Image Module

img

Object Module

object, param

Metainformation Module

meta

Link Module

link

Base Module

base

(*) = diese Module müssen bei XHTML Basic 1.0 mindestens unterstützt werden.

XHTML Basic 1.0 konformes [Beispiel](#)

8.4. XHTML 1.1 - Modul basiertes XHTML

Neuordnung von striktem XHTML 1.0 mit Hilfe von Modulen. Der Dokumenttyp ist

```
<!DOCTYPE
html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Definierte Module

Structure Module*

body, head, html, title

Text Module*

abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var

Hypertext Module*

a

List Module*

dl, dt, dd, ol, ul, li

Object Module

object, param

Presentation Module

b, big, hr, i, small, sub, sup, tt

Edit Module

del, ins

Bidirectional Text Module

bdo

Forms Module

button, fieldset, form, input, label, legend, select, optgroup, option, textarea

Table Module

caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr

Image Module

img

Client-side Image Map Module

area, map

Server-side Image Map Module

Attribut ismap von img

Intrinsic Events Module

Event Attribute

Metainformation Module

meta

Scripting Module

noscript, script

Stylesheet Module

style element

Style Attribute Module *Deprecated*

style attribute

Link Module

link

Base Module

base

Ruby Annotation Module

ruby, rbc, rtc, rb, rt, rp

Weitere Änderungen gegenüber XHTML 1.0 Strict sind: lang wird ersetzt durch xml:lang und name wird ersetzt durch id.

XHTML 1.1 konforme Beispiele: [basic](#), [alles](#).

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:35:56 CEST 2004

9. Cascading Style Sheets (CSS)

- Einordnung
 - Grundkonzepte von CSS
 - Layout mit CSS1
 - Beispiele und Hinweise
-

9.1. Einordnung

Layouts für Web-Publishing

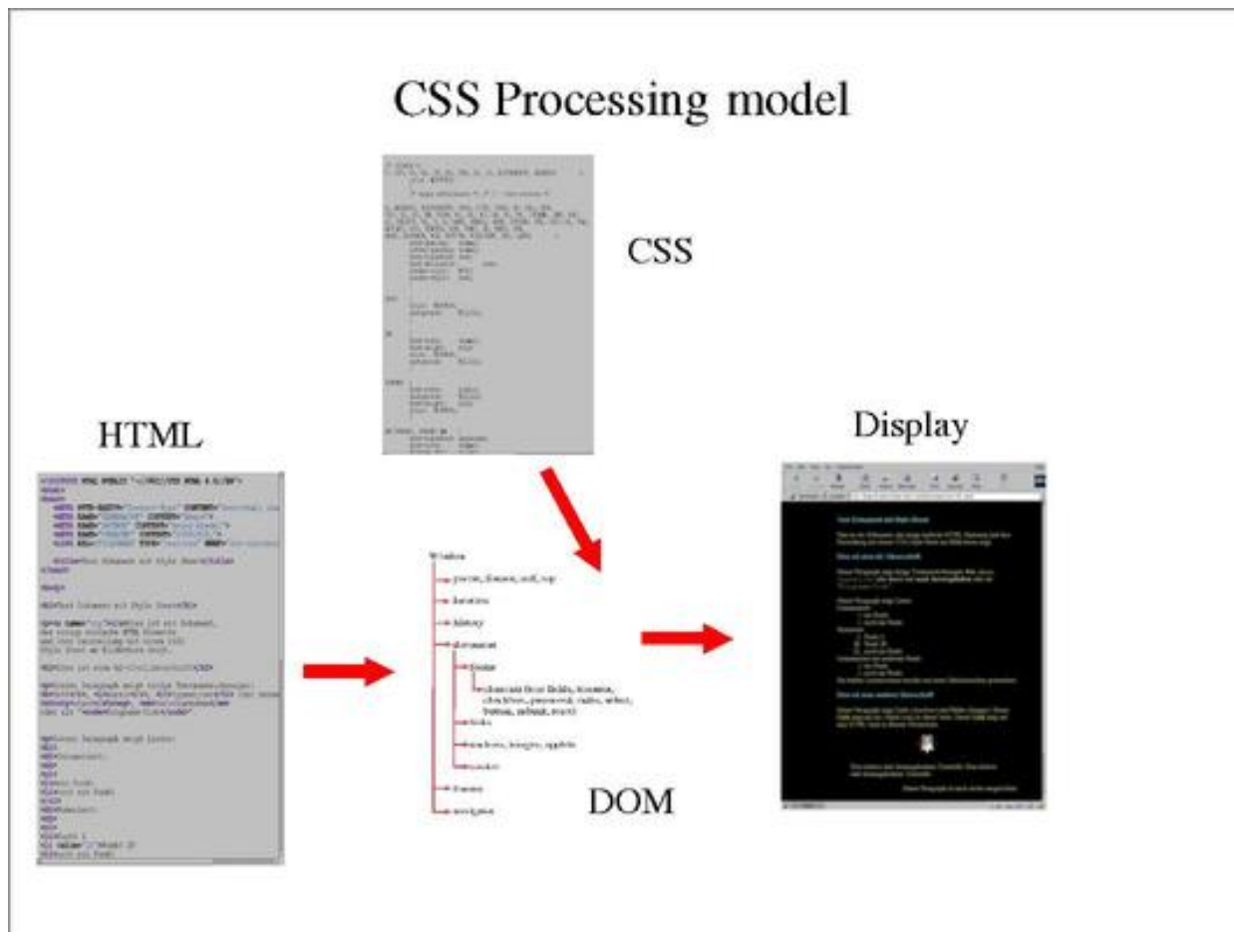
Trennung von Struktur und Präsentation wird möglich



- CSS1, CSS2
Cascading Style Sheets, Level 1 & Level 2
zu HTML
- DSSSL
Document Style Semantics and Specification Language
zu SGML
- XSL
Extensible Style Language
zu XML

Stand der Entwicklung:

- CSS1: W3C Recommendation, Dezember 1996
- CSS2: W3C Recommendation, Mai 1998
- CSS3: W3C Proposed Recommendation, 1999
- Browser wie die von Netscape oder Microsoft implementieren nur ca. 30% bis 80% der Spezifikation korrekt
- aber alle Hersteller haben sich zur Unterstützung von CSS1 verpflichtet



9.2. Grundkonzepte von CSS

- CSS Regeln:

```
element { property: value }  
body { background-color: white }
```

- Selektoren und Deklarationen
- Eigenschaften (properties) und Werte
- Regeln in eigener Datei oder in HTML eingebettet
- Gruppierung:

```
h1, h2, h3 { font-family: helvetica }  
h4 { color: red;  
    font-size: 8pt;  
}
```

- Kontext:

```
h1 { color: red }
```

Cascading Style Sheets (CSS)

```
h1 em      { color: purple }  
h1 em strong { color: fuchsia }
```

- Vererbung:

```
body { font-family: helvetica }  
h1   { color: red;  
      font-size: 14pt;  
    }
```

z.B. Schriftfamilie wird auf enthaltene Elemente vererbt

- Cascadierung:

```
em { font-style: italic }  
p em { font-weight: bolder }  
strong em { font-weight: bold }  
h1 em { font-style: normal }
```

- CSS ist "Fehlertolerant" wie HTML
- Gross-/Kleinschreibung wird ignoriert

Elemente für die kein Stil definiert wird, werden mit dem Browser Default-Stil dargestellt.

Verknüpfung mit HTML

- im link Element:

```
<link rel="stylesheet" type="text/css"  
      href="http://host/path/file.css">
```

- im style Element:

```
<style type="text/css">  
h3 { color: lime }  
</style>
```

- mit @import:

```
<style type="text/css">  
@import url(http://style.org/company.css)  
h1 { color: lime }  
</style>
```

- als Attribut:

```
<p style="color: red" > roter Text </p>
```

dies sollte man nicht machen

Element-Typen

Block-Elemente

blockquote, br, dd, dl, div, dt, hi, hr, html, li, object, ol, p, pre, ul

Inline-Elemente

a, em, i, img, span, strong, tt

Unsichtbare Elemente

link, meta, style, title

Auswahl mit Bezeichnern

- class, mehrfachverwendbare Bezeichner:

```
h1.slide { color: lime }
p.slide { color: green }
.buch1 { color: yellow; background-color: black }

<h1 class="slide">BlaBla</h1>
<h1 class="buch1">BluBlu</h1>
```

- id, eindeutige Bezeichner:

```
#sp3 { color: aqua }

<h1 id="sp3">BlaBla</h1>
```

- Pseudo-Klassen:

```
a:link { color: red } /* unvisited */
a:visited { color: blue }
a:active { color: red }
a:hover { color: purple } /* mouseover */
```

- Pseudo-Elemente:

```
p:first-line { color: fuchsia }
p:first-letter { color: red; background-color: lime }
```

- Sehr wichtige Definition:

```
h1.slide { color: fuchsia !important }
```

Algorithmus der Cascadierung

Es wird die anzuwendende Element-Eigenschaft Kombination gesucht:

1. Element entsprechend dem Selektor oder Vererbung
2. entsprechend explizitem Gewicht: !important geht vor
3. entsprechend dem Ursprung: Autor vor Benutzer vor Browser
4. entsprechend dem Detail-Level (specificity):
umgekehrt lexikographisch: (IDs, CLASSes, TAGnumber)

```
li { prop: .1. } --> (0,0,1)
ul li { prop: .2. } --> (0,0,2)
ul ol li { prop: .3. } --> (0,0,3)
li.rum { prop: .4. } --> (0,1,1)
ul li.rum { prop: .5. } --> (0,1,2)
ul.uni li.rum { prop: .6. } --> (0,2,2)
#hpc { prop: .7. } --> (1,0,0)
```

```
<li id="hpc" > .7. gewinnt </li>
<li class="rum"> .4., .5. oder .6. gewinnt </li>
<li > .1., .2. oder .3. gewinnt </li>
```

5. entsprechend der Reihenfolge

Beispiel: Header von HTML 2.0 in CSS1

```
h1, h2, h3, h4 { margin-top: 1em; margin-bottom: 1em }
h5, h6 { margin-top: 1em }
h1 { text-align: center }
h1, h2, h4, h6 { font-weight: bold }
h3, h5 { font-style: italic }

h1 { font-size: xx-large }
h2 { font-size: x-large }
h3 { font-size: large }
```

Algorithmus der Cascadierung in CSS2

Die Punkte 2 und 3 werden zu einem zusammengefasst.

1. entsprechend dem Ursprung: Autor vor Benutzer vor Browser
aber bei Angabe von !important gilt: Benutzer vor Autor vor Browser

9.3. Layout mit CSS1

Schriften

Liegen in verschiedenen Schnitten (Gestalt mit bestimmten Eigenschaften) vor. Die Schrift-Eigenschaften sind nicht einfach berechenbar (vergleiche TeX und Metafont).

- Schrift-Familie: font-family
Helvetica, Times, Western, Courier
sans-serif, serif, fantasy, monospace
- Schrift-Stil: font-style
normal, italic
- Schrift-Variante: font-variant
normal, small-caps
- Schrift-Gewicht: font-weight
normal, bold, lighter, 100, ..., 900 (sehr fett)
- Schrift-Grösse: font-size
absolut:
xx-small, x-small, small, medium, large, x-large, xx-large
12pt, 18pt, 10px, 1.0cm, 1.0in
relativ:
150%, 0.5em (Höhe M), 0.7ex (Höhe x)
- Schrift: font: style variant weight size/height family
zusammenfassende Deklaration

Beispiel

```
p { font-family: monospace;
    font-size: x-large;
}
```

So siehts aus.

Beispiele font-family:

Times, Helvetica, Verdana, Western, Courier, Zapf Chancery,

serif, sans-serif, monospace, cursive, fantasy.

Beispiele font-style:

normal, oblique, italic.

Beispiele font-variant:

normal, Small-Caps.

Beispiele font-weight:

normal, bold, bolder, lighter, 100, ..., 400, ..., 900.

Beispiele font-size:

xx-small, x-small, small, medium, large, x-large, xx-large,

2.0cm, 1.0in, 14pt, 34pt, 30px,

120%, 2.0em, 1.5ex.

Farben

- RGB-Farbmodell: Rot, Grün, Blau
bei Computer Bildschirmen
- CMYK-Farbmodell: Cyan, Magenta, Yellow, Black
bei Farbdruckern
- Farben:
black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua
rgb(rot,grün,blau), Werte: 0 <= rot, grün, blau <= 255, oder 0% <= rot, grün, blau <= 100%
#rgb, oder #rrggbb, Werte: 0 <= r, g, b <= F
- Schriftfarbe: color
- Hintergrundfarbe: background-color
- Hintergrundbild: background-image
url(http://host/pfad/bild)

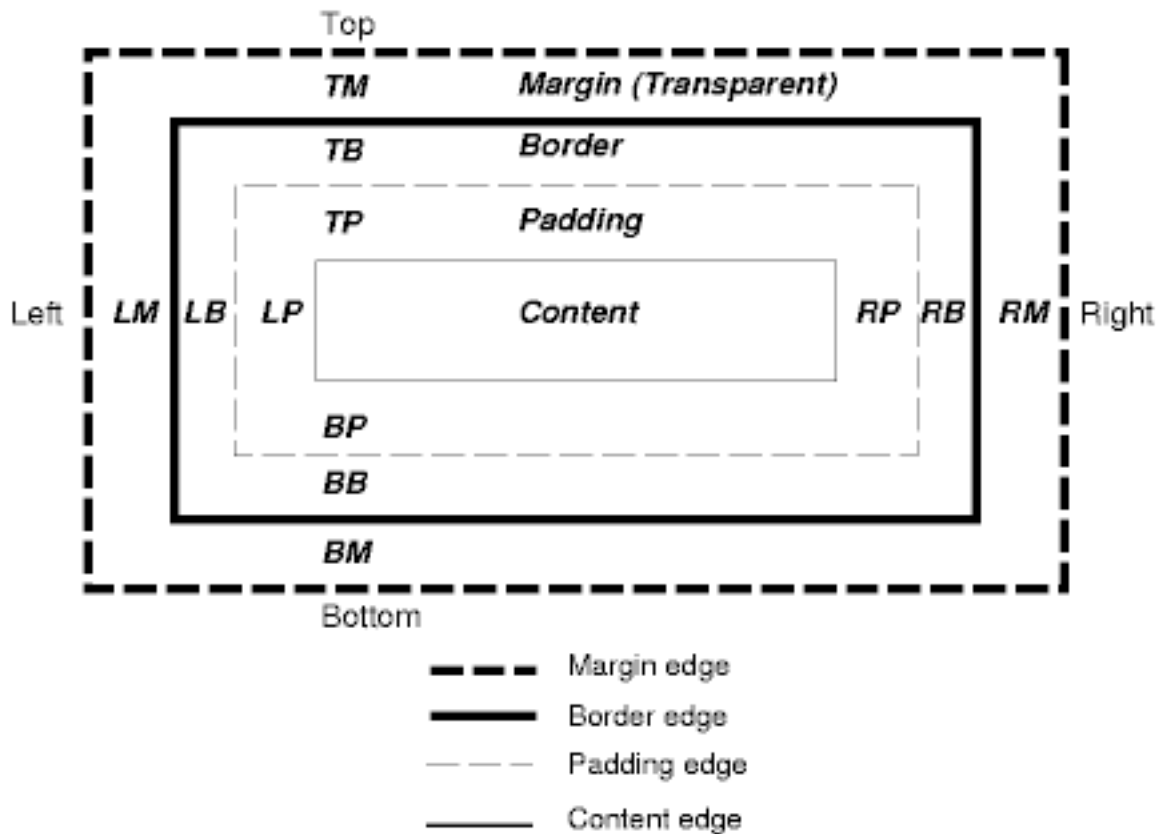
```
p { color: blue }
```

Schwarz, Hellgrau, Grau, Weiß, Dunkelrot, Rot, Purpur, helles Purpur, Grün, Hellgrün, dunkles Grün, Gelb, Dunkelblau, Blau, Blaugrün, helles Grünblau (Cyan).

weitere Eigenschaften

Box-Eigenschaften: Füllung (padding), Rahmen (border), Rand (margin)

```
Box mit was drin
```



Quelle: W3C, CSS2 Spezifikation

- Box-Eigenschaften:
margin,
margin-top, margin-right, margin-bottom, margin-left
padding,
border, border-color, border-style
width, height
- Text-Eigenschaften:
word-spacing,
white-space: pre, nowrap,
text-decoration: underline, blink,
text-transform: capitalize, uppercase, lowercase
- Text-Ausrichtung:
vertical-align, text-align,
text-indent, line-height
- Hintergrundbilder:
background-image: url(...)
background-repeat, background-attachment,
background-position

9.4. Beispiele und Hinweise

Beispiele vom W3C

HTML 2.0 in CSS1:

```
/* Copyright (c) 1998 W3C */
BODY {
  margin: 1em;
  font-family: serif;
  line-height: 1.1;
  background: white;
  color: black;
}

H1, H2, H3, H4, H5, H6, P, UL, OL, DIR,
MENU, DIV, DT, DD, ADDRESS, BLOCKQUOTE,
PRE, BR, HR { display: block }

B, STRONG, I, EM, CITE, VAR, TT, CODE, KBD, SAMP,
IMG, SPAN { display: inline }

LI { display: list-item }

H1, H2, H3, H4 { margin-top: 1em; margin-bottom: 1em }
H5, H6 { margin-top: 1em }
H1 { text-align: center }
H1, H2, H4, H6 { font-weight: bold }
H3, H5 { font-style: italic }

H1 { font-size: xx-large }
H2 { font-size: x-large }
H3 { font-size: large }

B, STRONG { font-weight: bolder } /*relative to the parent*/
I, CITE, EM, VAR, ADDRESS, BLOCKQUOTE { font-style: italic }
PRE, TT, CODE, KBD, SAMP { font-family: monospace }

PRE { white-space: pre }

ADDRESS { margin-left: 3em }
BLOCKQUOTE { margin-left: 3em; margin-right: 3em }

UL, DIR { list-style: disc }
OL { list-style: decimal }
MENU { margin: 0 } /* tight formatting */
LI { margin-left: 3em }

DT { margin-bottom: 0 }
DD { margin-top: 0; margin-left: 3em }

HR { border-top: solid }

A:link { color: blue } /* unvisited link */
A:visited { color: red } /* visited links */
A:active { color: lime } /* active links */

/* setting the anchor border around IMG elements
   requires contextual selectors */
A:link IMG { border: 2px solid blue }
A:visited IMG { border: 2px solid red }
A:active IMG { border: 2px solid lime }
```

Beispiele für Basis-HTML

[Beispiel](#) mit einfachen Hervorhebungen.

Stile aus der W3C Style Gallery

- Alle Stile in einem Frameset: [Multi](#)

- "Chocolate" Stil: [HTML](#), [CSS](#).
- "Midnight" Stil: [HTML](#), [CSS](#).
- "Modernist" Stil: [HTML](#), [CSS](#).
- "Oldstyle" Stil: [HTML](#), [CSS](#).
- "Steely" Stil: [HTML](#), [CSS](#).
- "Swiss" Stil: [HTML](#), [CSS](#).
- "Traditional" Stil: [HTML](#), [CSS](#).
- "Ultramarine" Stil: [HTML](#), [CSS](#).

Beispiele aus dem Web

- W3C Style Gallery:
<http://www.w3.org/StyleSheets/Core/>
- Genug ist Genug:
by [Stephen Traub](#)
- Microsoft CSS Gallery:
[Entrance](#)

Hinweise zur effektiven Nutzung von CSS

Alles was mit CSS1 machbar ist, ist in HTML 4.0 "deprecated", d.h. nicht mehr empfohlen, d.h. zur Nicht-Benutzung empfohlen.

- Benutze nur wenige zentrale Stile, am Besten ein Stil für alle Web-Seiten.
- Benutze nur gelinkte Stildateien.
- Benutze nur in Ausnahmefällen zusätzliche spezielle Stildateien.
- Lasse den Stil von einem Experten designen.
- Propagiere den Stil, biete gute Dokumentation.
- Beachte, dass die Web-Seiten auch noch gut aussehen, falls der Browser (UA) kein CSS kann.
- Benutze nicht mehr als *zwei* Fonts.
- Benutze relative Fontgrößen (200%,50%).
- Lasse !important für den Leser.

Beschränkungen von CSS1

Eine Übersicht über den Grad der Unterstützung von CSS(1) befand sich bis 2003 zum Beispiel bei <http://webreview.com/style/> unter dem Punkt 'Master list'. Die Liste ist heute (2004) nicht mehr notwendig und auch nicht mehr auffindbar.

- keine freie Positionierung
- keine volle Autoren-Kontrolle
- keine mehrfachen Spalten
- keine mehreren Schichten (Layers)
- Drucken verbesserungsfähig
- keine Sprachausgabe-Layouts

10. Cascading Style Sheets 2 (CSS2)

- CSS Level 2
 - Ausblick
-

10.1. CSS Level 2

- Positionierung
- Medientypen: Sprache, Drucker, Folien
- Schriftauswahl, Download
- Auto-Numerierung
- Tabellen

Beispiel: [Style Sheet für HTML 4.0 in CSS2](#)

Positionierung

Ab Netscape 4.x, IE 4.0, 5.0 implementiert

CSS-P = CSS1 + Positionierung

- Box-Position:
`position: static, relative, absolute, fixed`
 - static** =
normaler Elementfluss
 - relative** =
zunächst wie static, dann relativ zu dieser Position
 - absolute** =
unabhängig vom Elementfluss an einer festen Stelle im *Dokument*
 - fixed** =
unabhängig vom Elementfluss an einer festen Stelle im *Fenster*
- `left: length`
- `top: length`
- `height: length`
- `width: length`
- `z-index: integer`
- `visibility: hidden, visible`

Animation mit [HTML, CSS, JavaScript und DOM](#)

Dateien: [HTML](#), [CSS](#), [JavaScript](#).

Medientypen

- screen: Bildschirmausgabe
- print: Druckausgabe
- aural: Sprachausgabe
- braille: Blindenschrift
- embossed: Blindenschrift-Drucker
- handheld: Mobilfunk, Handy
- projection: Projektor, Video-Beamer
- tty: Text-Bildschirm
- tv: TV-Bildschirm

Beispiel:

```
@media print {  
    body { font-size: 10pt }  
}  
@media screen {  
    body { font-size: 12pt }  
}
```

Mediengruppen

- continous oder paged
- visual oder aural oder tactile
- grid oder bitmap
- interactive oder static

neue Selektoren

für Attribute

```
p[align]          { font-style: italic; }  
p[align="center"] { font-style: italic; align: center; }  
p[align~="center"] { font-style: italic; align: center; }
```

für Kinder und nebeneinanderstehende Elemente

```
p > em           { font-style: italic; }  
h1 + h2          { margin-top: -5mm; }
```

für Sprachauswahl

```
p:lang(de)       { font-family: sans-serif; }  
p:lang(fr)       { font-family: serif; }
```

neue Pseudo-Elemente

```
h1:before        { content: counter(kapitel) ". "; }  
p:after          { content: " © BWL 1999"; }
```

neue Display-Typen

```
h3      { display: run-in; }
dt      { display: compact; }
tr      { display: table-row; }
```

Fliesstext

```
img      { float: left; }
img.x    { float: right; }
p        { clear: left; }
```

Erweiterungen bei Fonts

- `font-stretch`: Dehnungsverhalten
normal, wider, narrower, condensed, expanded
- `font-size-adjust`: Angleichung der Zeichengrößen zwischen verschiedenen Fonts, Aspekt-Wert
- `@font-face` Beschreibung und Auswahl von Fonts

```
@font-face {
    font-family: "Robson Celtic";
    src: url("http://host/fonts/rob-celt");
}

p.rc { font-family: "Robson Celtic"; }
```

Fontauswahl

- Auswahl per Fontname
Problem: es gibt kein anerkanntes Namensschema
- Auswahl "ähnlicher" Fonts (intelligent font matching)
- Erzeugung von Fonts (font synthesis)
- Download von Fonts

@font-face

- alle Eigenschaften von Fonts können verwendet werden
- `font-family`, `font-style`, `font-variant`, `font-weight`, `font-stretch`, `font-size`
- zusätzlich kommen neue Deskriptoren hinzu:
- `uni-code-range`: Bereich vorhandener Zeichen (glyphs)
U+0-7FFFFFFF
- `units-per-em`: Anzahl der Einheiten per 'em'
- `src`: URL (zum Download) oder Bezeichnung für einen Font

```
src: local("Verdana"), url("../fonts/verdana") format("type-1")
```

- Deskriptoren für Fontauswahl
`panose-1`: 0 0 0 0 0 0 0 0 0 0
Panose-1 Nummer
`stemv`: vertikaler Stem-Wert ("M", em, Höhe des grossen M)
`stemh`: horizontaler Stem-Wert ("x", ex, Höhe des kleinen x)
`slope`: Neigungswinkel
`cap-height`: Grösse der Grossbuchstaben
`x-height`: Grösse der Kleinbuchstaben
`ascent`: Grösse der Buchstaben ohne Akzente
`descent`: Grösse der Buchstaben ohne untere Akzente

- Deskriptoren für Fontsynthese
width: Breite von Buchstaben
bbox: maximaler Umriss von Buchstaben
definition-src: definiert wo die Spezifikation zu finden ist
- Deskriptoren für Ausrichtung von Fonts untereinander
baseline: untere Basisline der Schrift
centerline: Mittellinie der Schrift
mathline: Ausrichtung der mathematischen Zeichen
topline: obere Basisline der Schrift
- die Auswahl geeigneter Fonts erfolgt durch einen (längeren) Algorithmus

Beispiel

```
@font-face {  
    font-family: "Swiss 721 Condensed";  
    src: url("http://host/fonts/swiss721co.pfr");  
    font-style: normal, italic;  
    font-stretch: condensed;  
}  
  
p.sc { font-family: "Swiss 721 Condensed"; }
```

Zähler und Textersetzungen

- Pseudo-Elemente :before, :after
- Textersetzungen content:
"string", URL, counter(cnt), attr(X), open-quote, close-quote
- Initialisieren von Zählern counter-reset: cnt
- Weiterzählen counter-increment: cnt
- Zähler-Format counter(name, type)
disk, circle, square, upper-latin, hebrew, upper-roman
- Einrückungen für Zähler marker-offset:

```
@media aural {  
    blockquote { content: url("bq-music.wav"); }  
}  
  
h1:before {  
    content: "Kapitel " counter(kapitel) ". ";  
    counter-increment: kapitel;  
    counter-reset: abschn;  
}  
  
h2:before {  
    content: counter(kapitel) "." counter(abschn) ". "; }  
    counter-increment: abschn;
```

Stile für Drucker

- Druckseiten @page { ... }
- Seitengröße size:
- Ränder margin:
- für linke und rechte Seiten :left, :right
- Seitenumbruch page-break-before:
page-break-after:
page-break-inside:
auto, always, avoid, left, right

- Seitenumbruch in Paragraphen
Anzahl Zeilen am Fuss orphans: int
Anzahl Zeilen am Kopf widows: int

```
@media print {  
  @page {  
    size: auto;  
    margin: 10%;  
  }  
  @page :right {  
    margin-left: 10;  
    margin-right: 5;  
  }  
  @page :left {  
    margin-left: 5;  
    margin-right: 10;  
  }  
}
```

Stile für Tabellen

- zugeschnitten für das HTML 4.0 Tabellenmodell
- rechteckige Anordnung von Zellen
- zeilenweise Anordnung

| Beispiel für Tabelle | | |
|----------------------|----------|----------|
| Spalte 1 | Spalte 2 | Spalte 3 |
| 1 | 2 | 3 |
| 4 | | 6 |

```
<table class="exam" border="1"  
  cellpadding="10" cellspacing="10"  
  summary="HTML Elemente">  
<tr>  
<th>Spalte 1</th><th>Spalte 2</th><th>Spalte 3</th>  
</tr>  
<tr>  
<td>1</td><td>2</td><td>3</td>  
</tr>  
<tr>  
<td>4</td><td></td><td>6</td>  
</tr>  
<caption>Beispiel für Tabelle</caption>  
</table>
```

mit folgendem Style Sheet

```
table.exam {  
  font-size: large;  
  background-color: lime; }  
table.exam td {  
  text-align: center;  
  color: red; }  
table.exam caption {  
  caption-side: right;  
  color: blue; }
```

freidefinierbare Tabellen-Elemente

Neue display: Werte ermöglichen Tabelleneigenschaften für beliebige Elemente, wichtig für XML (Auswahl).

- table wie table in HTML 4.0
- table-row wie tr
- table-column wie col
- table-cell wie td, th
- table-caption wie caption

```
table { display: table }
tr { display: table-row }
th, td { display: table-cell }
col { display: table-col }
caption { display: table-caption }
```

weitere Tabellen-Eigenschaften

- Grundlayout
table-layout: auto | fixed
- Ausrichtung an bestimmten Zeichen
text-align: "."
- Rand-Modelle
border-collapse: collapse, separate
- Ränder
border-spacing: len
- leere Zellen
empty-cells: show, hide
- Randstil
border-style: solid, dotted, groove, ridge
- Sprachausgabe
speak-header: once, always

```
td { text-align: "," }
td:before { content: "€" }
table { empty-cells: show }
```

Stile für Sprachausgaben

Eigenschaften: Umgebung-/Raumabhängigkeit, Zeitabhängigkeit, Sprachqualität.

- Lautstärke
volume: soft | loud | x%
- Sprechart
speak: normal | spell-out
- Pausen
pause-before: zeit
pause-after: zeit
pause: before after
- "Ton-Icons" = Cue
cue-before: url(.)
cue-after: url(.)
cue: before after
- Gleichzeitiges Abspielen
play-during: url(.) mix? repeat?
- Räumliche Eigenschaften
azimuth: Raumseitenwinkel, behind, left, right
elevation: Raumhöhenwinkel, below, above

- weitere Eigenschaften
speech-rate: slow | fast, Worte pro Minute (180-200),
voice-family: male | female | 'speaker',
pitch: freq, Frequenz (120Hz)
pitch-range: ,
stress: int, Betonung
richness: int, "Fülle"
speak-punctuation: code | none,
speak-numeral: digits | continuous

```
@media aural {  
  h2      { pause: 30ms 40ms; }  
  a       { cue-before: url("a-bell.wav"); }  
  em      { play-during: url("em-sound.wav") mix repeat; }  
  p.note  { azimuth: behind; }  
  p.dog   { elevation: below; }  
}
```

Weitere Änderungen gegenüber CSS1

- Berücksichtigung der Text-Schreibrichtung
direction: ltr|rtl
- Behandlung von Text-Ausschnitten
clip: shape,
overflow: scroll
- Cursor Darstellung
cursor: crosshair | text | wait | url(.)

Beispiele

[Beispiel](#) mit CSS Level 2 Konstrukten.

10.2. Ausblick

CSS3

- Paged Media Support, z.B. WAP auf Handys
- DOM Level 2 Support
- Scalable Vector Graphics (SVG)
- User Interface, z.B. "Kiosk" Mode
- International Layout, z.B. Arabisch, Japanisch
- Multicolumn Layout

XSL

- CSS und XML
 - XSL
 - XSLT
-

Last modified: Sat May 22 12:36:17 CEST 2004

11. JavaScript



Button: JavaScript

- Einleitung
- Sprachkonstrukte
- Anwendungen

11.1. Einleitung

- JavaScript
- VBScript
- ECMA-Script

Entwicklung

- LiveScript
- Version 1.0 zunächst in Netscape 2.0
- jetzt auch in MS Internet-Explorer
- JScript entspricht etwa Version 1.0
- aktuelle Version 1.2
- Standardisierung als ECMA-Script (European Computer Manufacturer Association)

Überblick

- Interpretierte Sprache in Web Browsern
- Syntax ähnlich C++, Java, "objektbasiert"
- Programtext wird in HTML-Seiten eingebettet
- Erweiterungen zur Interaktion mit dem Browser
Mouse-Clicks, Form-Input, Seitennavigation

Einbettung in HTML

Verwendung von JavaScript durch Einbettung in HTML Seiten.

```
<script type="text/javascript" language="JavaScript">
<!-- to hide script contents from old browsers
... JavaScript ...
// end hiding contents from old browsers -->
```

```
</script>
```

oder auch:

```
<script type="text/javascript" language="JavaScript" src="vorles.js"></script>
```

Benutzung von JavaScript Funktionen in HTML Tags.

```
<input type=... value=...  
  onclick="jsfunc('arguments');">
```

Kurzes Beispiel

Ausgabe von "Hallo ..." in einer HTML Seite.

```
<html>  
<head>  
</head>  
<body>  
<script type="text/javascript" language="JavaScript">  
  document.write("Hallo von JavaScript !")  
</script>  
  Das war's auch schon.  
</body>  
</html>
```

Das war's auch schon.

11.2. Sprachkonstrukte

Syntax ähnlich C++, Java, "objektbasiert"

Werte und Variablen

- Konversion von Zahlen in Strings falls ein Operand ein String.
- Umkehrung nur mit speziellen Funktionen: `parseInt`, `parseFloat`, `eval`.
- Keine Deklarationspflicht für Variablen. `var x = "Hallo !\n"`
- Operatoren und Ausdrücke wie in C. `y += x--`
- Verwendung in HTML: `width="{JSvar}";`

Kontrollstrukturen

- Statements, Expressions, { Statement-Folge }
- if-Statement

```
if (condition) {  
  statements1  
[ else {  
  statements2 } ]
```

- for-Statement

```
for ([initial-expression]; [condition];  
  [increment-expression]) {  
  statements  
}
```

JavaScript

- while-Statement

```
while (condition) {  
    statements  
}
```

- for-in-Statement

```
for (variable in object) {  
    statements }  
}
```

- Zum Beenden bzw. Abkürzen von Schleifen. break bzw. continue
- Es gibt einen Datentyp Boolean. toBoolean

Objekte

JavaScript Objekte

- Array
- Boolean
- Date
- Function
- Image
- Math
- Number
- String

Verwenden von Objekten

```
objectName.propertyName  
objectName['propertyName']
```

```
this  
this.propertyName
```

Funktionen

```
function funktionsName ( param1 [,param2] ...[,paramN] ) {  
    ...  
    return( ... );  
}
```

Beispiel:

```
function show_props(obj, obj_name) {  
    var result = "";  
    for (var i in obj)  
        result += obj_name + "." + i + " = " + obj[i] + "\n";  
    return result;  
}
```

Button: **show_props(document)**

Button: **show_props(window)**

Button: **show_props(window.navigator)**

Button: **show_props(document.forms)**

Eingabe: **Textfield:** **document** Button: **show_props(Eingabe)**

Konstruktor für Objekte.

```
function objectType ( param1 [,param2] ...[,paramN] ) {  
    this.property1 = param1;  
    ...  
    this.propertyN = paramN;  
}
```

Erzeugen von Objekten.

```
objectName = new objectType ( param1, ...[,paramN] )
```

Methoden und Funktionen.

```
objectName.methodName = function_name
```

```
objectName.methodName(params);
```

Prototypen.

```
new ObjectName();  
ObjectName.prototype.pName = wert;  
objectName = new ObjectName();  
x = objectName.pName;
```

Beispiel:

```
function alter() {  
    var today = new Date();  
    return( today.getFullYear() - this.baujahr );  
}  
  
function Auto(modell, baujahr) {  
    this.modell=modell;  
    this.baujahr=baujahr;  
    this.alter = alter;  
}  
  
ford = new Auto("Fiesta", 1995);  
document.write("Alter = ", ford.alter());
```

Es gibt diverse eingebaute Objekte und Funktionen. Öffnen und schließen von Windows, Alert Meldungen und Confirmations.

Event Handler

- Aufbau: **event (elemente)**
- onabort (image)
- onblur (window, frame, select, text, textarea)
- onchange (select, text, textarea)
- onclick (alles ausser: applet, base, basefont, bdo, br, font, frame, frameset, head, html, iframe, isindex, meta, param, script, style, title)
- onerror (image, window)
- onfocus (window, frame, select, text, textarea)
- onload (image)
- onmouseout (area, a)

- onmouseover (area, a)
- onreset (form)
- onselect (text, textarea)
- onsubmit (form)
- onunload (window)

Beispiel:

```
function validate(obj, lowval, hival) {  
  if ((obj.value < lowval) || (obj.value > hival))  
    alert("Value must be greater than " + lowval + " and less than " + hival + ".")  
}
```

```
<input type = "text" name = "age" size = "3"  
  onchange="validate(this, 18, 99)" />
```

Alter: **Textfield:**

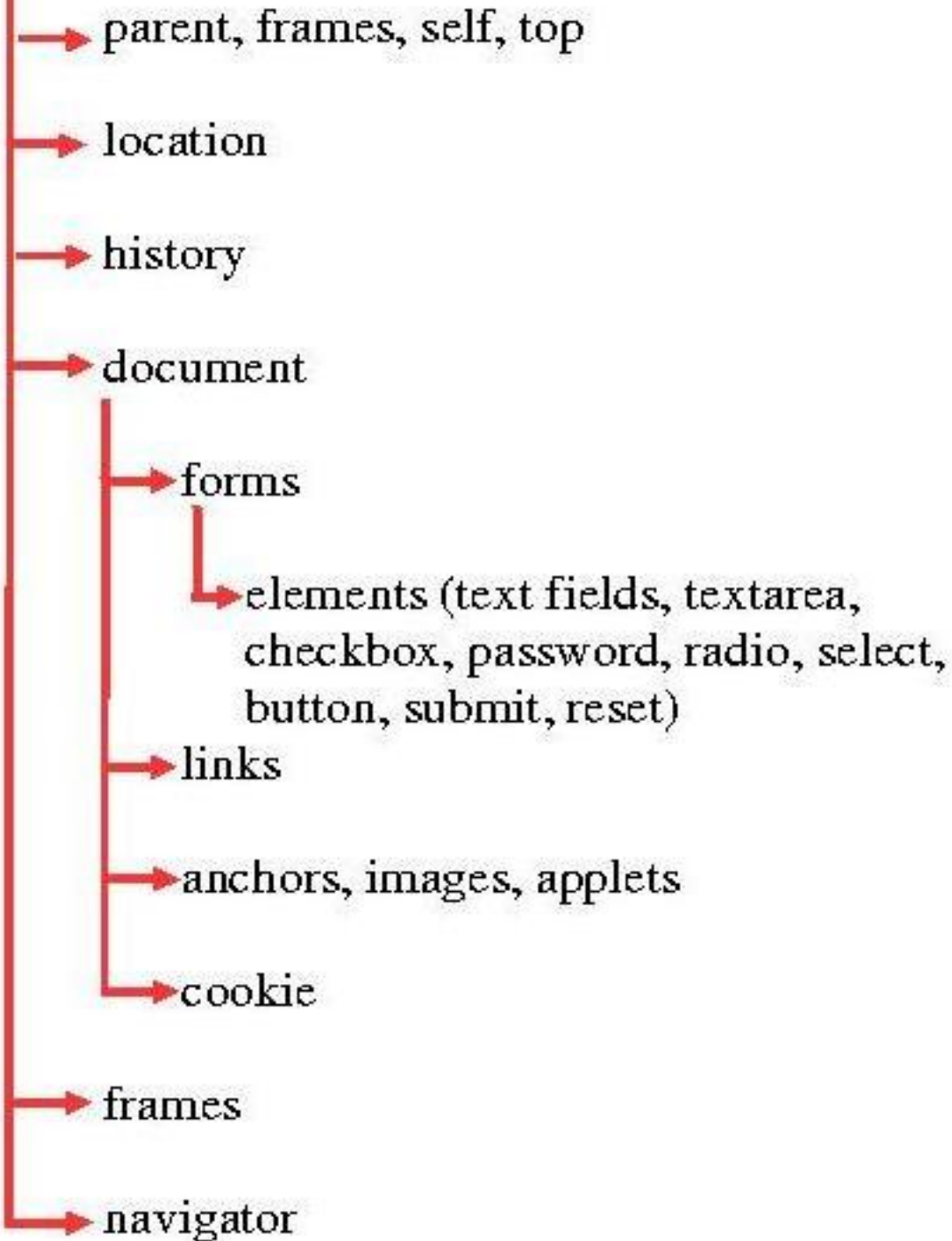
Bitte Alter eintragen und Fenster anklicken.

vordefinierte Objekte

- Document Object Model, DOM
- Browser Objekte

Objekthierarchie

Window



Beispiel für die Objekte eines Dokuments.

```
<title>A Simple Document</title>
<body>
<form name="myform" action="FormProc()" method="get" >
Enter a value:
<input type="text" name="text1" value="blahblah" size="20" >
Check if you want:
<input type="checkbox" name="check1" checked="checked"
      onclick="update(this.form)"> Option #1
<p>
<input type="button" name="button1" value="Press Me"
      onclick="update(this.form)">
</form>
</body>
```

Dann sind unter anderem folgende Objekte definiert.

```
document.title = "A Simple Document"

document.myform
document.myform.check1
document.myform.button1

document.myform.button1.value = "Press Me"
document.myform.button1.name = button1
```

Methoden vordefinierter Objekte

- w = window.open(url [,name,options])
- w.close()
- alert(mitteilung)
- jein = confirm(frage)
- antwort = prompt(frage)

11.3. Anwendungen

Feldprüfungen in Formularen

```
Feldinhaltsprüfungen in Formularen: <br>
<script type="text/javascript" language="JavaScript">
<!-- hide from strangers
function checkForm(frm) {
    if (frm.my_name.value.length > 0) return true
    else {
        alert("Please enter your name.")
        return false
    }
}
//-->
</script>
```

```
<form action="http://trumpf-2.rz.uni-mannheim.de/cgi-bin/ex2.cgi"
      onSubmit="return checkForm(this);">
Mein Name:
<input type="text" name="my_name" size="20">
<p>
Mein Status:
```

```
<input type="radio" name="my_status" value="student">Student  
<input type="radio" name="my_status" value="employee" checked>Mitarbeiter  
<input type="radio" name="my_status" value="professor">Professor  
<p>  
<input type="reset" value="Reset"> <input type="submit" value="Send">  
</form>  
</p>
```

In diesem [Formular](#) werden die Eingaben (hier nur vom Textfeld "my_text") auf Richtigkeit geprüft, bevor die Daten an das CGI-Script geschickt werden.

Debuging von JavaScript Programmen mit [javascript](#):

Bestimmung von [Unix Dateirechten](#)

Animation mit [HTML](#), [CSS](#), [JavaScript](#) und [DOM](#)

Dateien: [HTML](#), [CSS](#), [JavaScript](#).

Beispiele von [Gamelan](#).

11.4. Ausblick

- Sicherheit
- DOM
- ECMA-Script Components
- LiveConnect zu Java
- LiveWire
JavaScript auf der Server Seite

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:36:31 CEST 2004

12. Document Object Model (DOM)

- Einleitung
- DOM Core
- DOM HTML
- Bindungen
- Ausblick

12.1. Einleitung

- Document Objekt Modell von JavaScript
- Document Object Modell von Microsoft
- Definiert das Objekt-Modell von Web-Dokumenten d.h. Dokument als Datenstruktur in Programmen
- erlaubt den dynamischen Zugriff und die Modifikation von Inhalt und Struktur von Dokumenten
- Level 0: Durchschnitt dessen, das was in Netscape und MS IE implementiert ist
- Level 1: seit 1. Oktober 1998 W3C Recommendation
- Spezifikation besteht aus: Core, HTML und XML Teilen
- Programm-Schnittstellen sind *Plattform und Sprachneutral* mit OMG-IDL definiert
- Es gibt Bindungen zu JavaScript (ECMA-Script), Java und VBScript
- Level 2, Core, Style: W3C Recommendation, 13. November 2000
mit Ausnahme von HTML: W3C Recommendation, 9. Januar 2003.
- Level 3, Core: W3C Working Draft, 26. Februar 2003.
- nicht identisch mit *Dynamic HTML*

DOM Eigenschaften

- mit DOM kann jedes Element und dessen Inhalt in einem HTML (und XML) Dokument referenziert werden
- die Elemente, ihr Inhalt und ihre Struktur kann modifiziert werden
- geeignet für Script-Sprachen aber z.B. auch für HTML-Editoren
- die Erzeugung von Dokument Objekten ist nur mit Einschränkungen spezifiziert
- mit DOM Level 2 sind zusätzlich auch die Stilinformation (von CSS), Ereignisse (Events von JavaScript) referenzierbar und manipulierbar

Beispiel

Die Tabelle

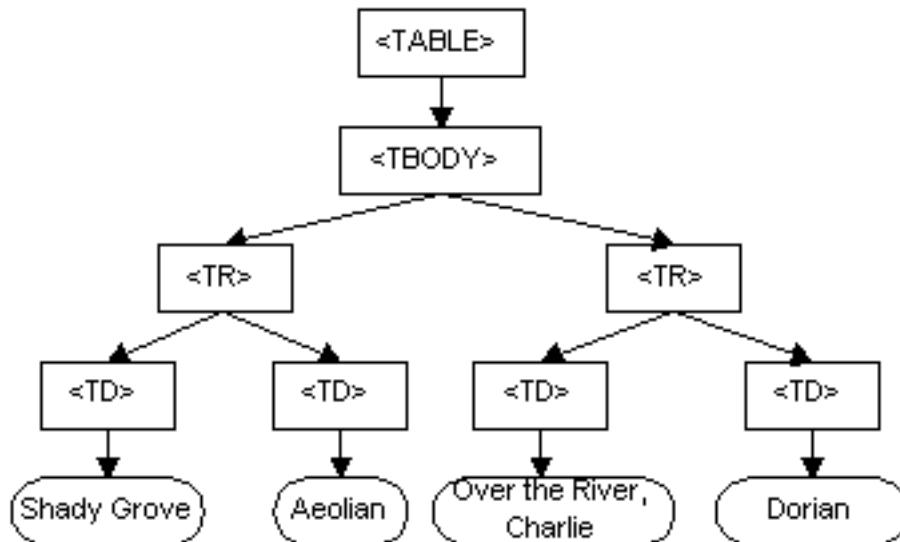
| | |
|-------------------------|---------|
| Shady Grove | Aeolian |
| Over the River, Charlie | Dorian |

in HTML

```
<table border="1">  
<tbody>  
<tr>
```

```
<td>Shady Grove</td>
<td>Aeolian</td>
</tr>
<tr>
<td>Over the River, Charlie</td>
<td>Dorian</td>
</tr>
</tbody>
</table>
```

sieht in DOM (bei einer Baum-Implementierung) wie folgt aus:



DOM Darstellung des Beispiels, Quelle W3C

Objekt Modell

- Beschreibung der Objekte und Schnittstellen, die zur Darstellung und Manipulation von Dokumenten notwendig sind.
- Beschreibung der Bedeutung (Semantik) der Schnittstellen und der Objekte, sowie deren Attribute und Verhalten.
- Beschreibung der Beziehungen und Interaktionen zwischen den Schnittstellen und Objekten.
- Mehr als eine reine Daten-Spezifikation, sondern auch eine Spezifikationen der den Daten (Objekten) zugehörigen Methoden.
- Keine Spezifikation der Bedeutung von HTML oder XML, DOM respektiert diese Semantik.
- In DOM Core keine Spezifikation von *Entities* als Objekte.

Interface Definition Language (IDL)

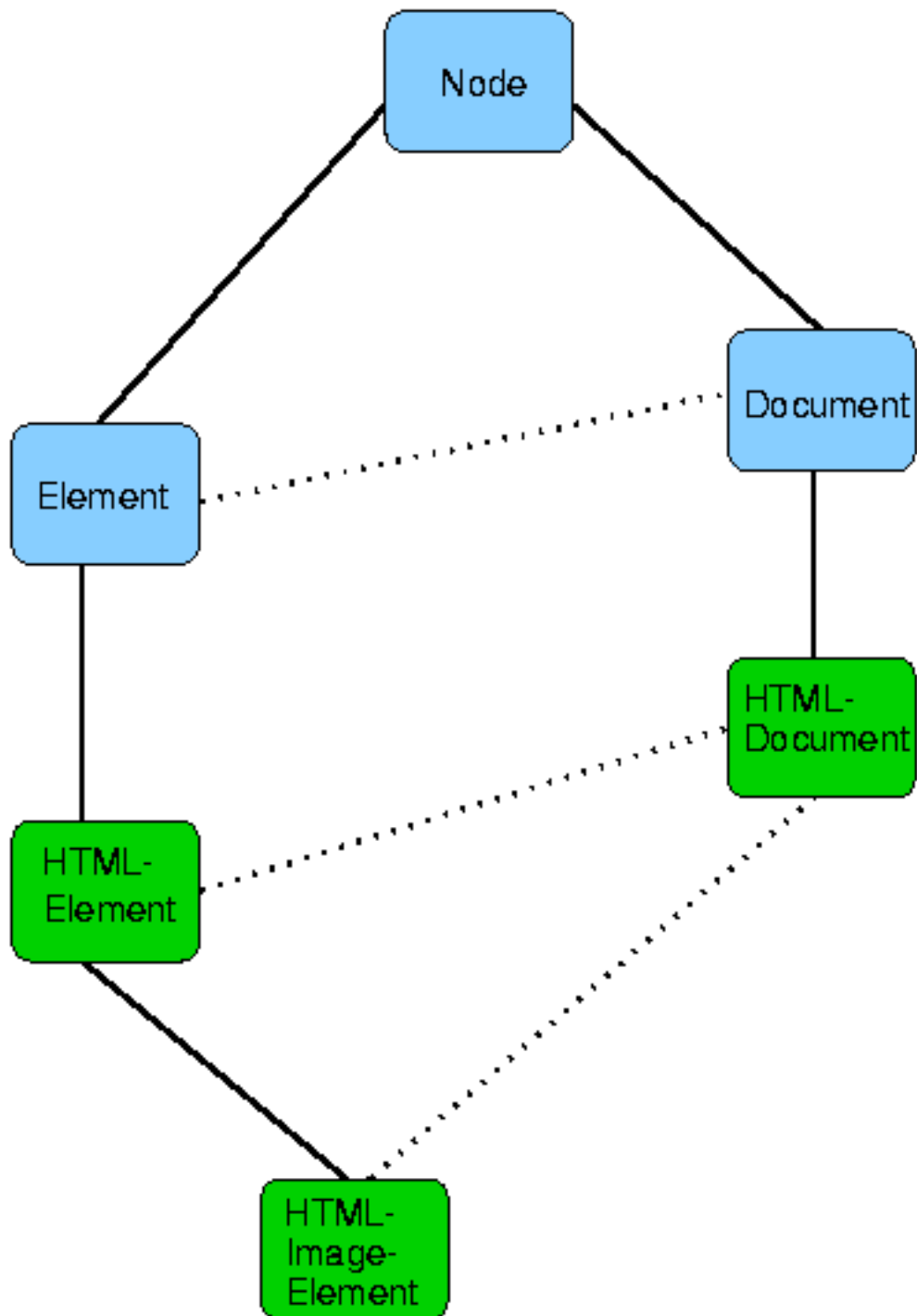
- Component Object Model (COM)
- Common Object Request Broker Architecture (CORBA)
- Object Management Group (OMG)

Unterschiede zu Implementierungen

- DOM ist unabhängig von einer bestimmten Implementierung.
- Auf Attribute kann nur über die implementierten `get()`/`set()` Methoden zugegriffen werden.
- Die Implementierung kann weitere Schnittstellen hinzufügen.

- DOM kennt keine Konstruktoren für Objekte
 - zur Erzeugung von Objekten XXX müssen die entsprechenden `createXXX()` Methoden der Document Klasse verwendet werden.
-

12.2. DOM Level 1, Core



DOM Interface Vererbung (---)
und Elementbeziehungen (...)

Notation: Abgeleitetes_Interface : Basis_Interface

Spezifiziert gemeinsame Basis für HTML und XML Teile:

- DOMString
- DOMException
- DOMImplementation
- DocumentFragment : Node
- Document : Node
- Node
- NamedNodeMap
- CharacterData : Node
- Attr : Node
- Element : Node
- Text : CharacterData

Interface: Node

```
interface Node {
  // NodeType
  const unsigned short ELEMENT_NODE           = 1;
  const unsigned short ATTRIBUTE_NODE         = 2;
  const unsigned short TEXT_NODE              = 3;
  const unsigned short CDATA_SECTION_NODE     = 4;
  const unsigned short ENTITY_REFERENCE_NODE  = 5;
  const unsigned short ENTITY_NODE            = 6;
  const unsigned short PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short COMMENT_NODE           = 8;
  const unsigned short DOCUMENT_NODE          = 9;
  const unsigned short DOCUMENT_TYPE_NODE     = 10;
  const unsigned short DOCUMENT_FRAGMENT_NODE = 11;
  const unsigned short NOTATION_NODE          = 12;

  readonly attribute DOMString      nodeName;
  attribute          DOMString      nodeValue;
  // raises(DOMException) on setting
  // raises(DOMException) on retrieval
  attribute          unsigned short nodeType;
  readonly attribute Node           parentNode;
  readonly attribute NodeList      childNodes;
  readonly attribute Node          firstChild;
  readonly attribute Node          lastChild;
  readonly attribute Node          previousSibling;
  readonly attribute Node          nextSibling;
  readonly attribute NamedNodeMap  attributes;
  readonly attribute Document      ownerDocument;
  Node               insertBefore(in Node newChild,
                                in Node refChild)
                                raises(DOMException);
  Node               replaceChild(in Node newChild,
                                in Node oldChild)
                                raises(DOMException);
  Node               removeChild(in Node oldChild)
                                raises(DOMException);
  Node               appendChild(in Node newChild)
                                raises(DOMException);
  boolean            hasChildNodes();
  Node               cloneNode(in boolean deep);
};
```

Interface: Document

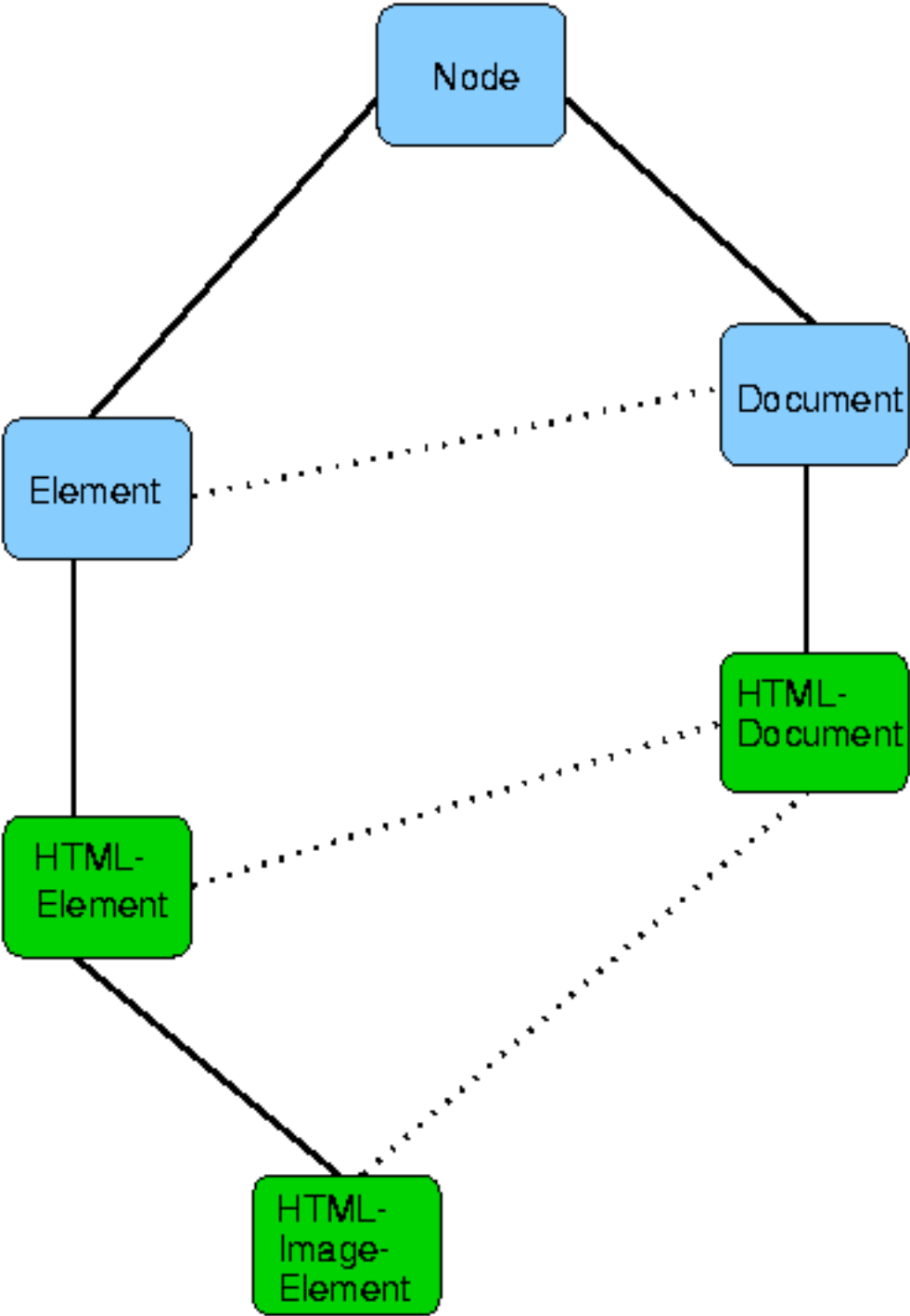
```
interface Document : Node {
```

```
readonly attribute DocumentType doctype;
readonly attribute DOMImplementation implementation;
readonly attribute Element documentElement;
Element createElement(in DOMString tagName)
    raises(DOMException);
DocumentFragment createDocumentFragment();
Text createTextNode(in DOMString data);
Comment createComment(in DOMString data);
CDATASection createCDATASection(
    in DOMString data)
    raises(DOMException);
ProcessingInstruction createProcessingInstruction(
    in DOMString target,
    in DOMString data)
    raises(DOMException);
Attr createAttribute(in DOMString name)
    raises(DOMException);
EntityReference createEntityReference(
    in DOMString name)
    raises(DOMException);
NodeList getElementsByTagName(
    in DOMString tagName);
};
```

Interface: Element

```
interface Element : Node {
    readonly attribute DOMString tagName;
    DOMString getAttribute(in DOMString name);
    void setAttribute(in DOMString name,
        in DOMString value)
        raises(DOMException);
    void removeAttribute(in DOMString name)
        raises(DOMException);
    Attr getAttributeNode(
        in DOMString name);
    Attr setAttributeNode(in Attr newAttr)
        raises(DOMException);
    Attr removeAttributeNode(
        in Attr oldAttr)
        raises(DOMException);
    NodeList getElementsByTagName(
        in DOMString name);
    void normalize();
};
```

12.3. DOM Level 1, HTML



DOM Interface Vererbung (---) und Elementbeziehungen (...)

Document Properties: **Button: show_props(document)**

Spezifiziert den HTML Teil:

- HTMLCollection
- HTMLDocument : Document
- HTMLElement : Element
- HTMLHeadElement : HTMLElement
- HTMLBodyElement : HTMLElement
- HTMLLinkElement : HTMLElement
- HTMLFormElement : HTMLElement
- HTMLInputElement : HTMLElement
- HTMLUListElement : HTMLElement
- HTMLParagraphElement : HTMLElement
- HTMLTableElement : HTMLElement

Interface: HTMLDocument

```
interface HTMLDocument : Document {
    attribute DOMString      title;
    readonly attribute DOMString referrer;
    readonly attribute DOMString domain;
    readonly attribute DOMString URL;
    attribute HTMLElement    body;
    readonly attribute HTMLCollection images;
    readonly attribute HTMLCollection applets;
    readonly attribute HTMLCollection links;
    readonly attribute HTMLCollection forms;
    readonly attribute HTMLCollection anchors;
    attribute DOMString      cookie;
    void open();
    void close();
    void write(in DOMString text);
    void writeln(in DOMString text);
    Element getElementById(
        in DOMString elementId);
    NodeList getElementsByName(
        in DOMString elementName);
};
```

Interface: HTMLElement

```
interface HTMLElement : Element {
    attribute DOMString id;
    attribute DOMString title;
    attribute DOMString lang;
    attribute DOMString dir;
    attribute DOMString className;
};
```

Interfaces zu Link, UL, LI, P, IMG

```
interface HTMLLinkElement : HTMLElement {
    attribute boolean disabled;
    attribute DOMString charset;
    attribute DOMString href;
    attribute DOMString hreflang;
    attribute DOMString media;
    attribute DOMString rel;
    attribute DOMString rev;
    attribute DOMString target;
```

```

};      attribute  DOMString      type;

interface HTMLUListElement : HTMLElement {
    attribute  boolean      compact;
    attribute  DOMString    type;
};

interface HTMLLIElement : HTMLElement {
    attribute  DOMString    type;
    attribute  long        value;
};

interface HTMLParagraphElement : HTMLElement {
    attribute  DOMString    align;
};

interface HTMLImageElement : HTMLElement {
    attribute  DOMString    lowSrc;
    attribute  DOMString    name;
    attribute  DOMString    align;
    attribute  DOMString    alt;
    attribute  DOMString    border;
    attribute  DOMString    height;
    attribute  DOMString    hspace;
    attribute  boolean      isMap;
    attribute  DOMString    longDesc;
    attribute  DOMString    src;
    attribute  DOMString    useMap;
    attribute  DOMString    vspace;
    attribute  DOMString    width;
};

```

12.4. Bindungen für ECMA-Script

| IDL Bezeichnung | ECMA-Script |
|-----------------|-----------------------|
| interface | Object |
| attribute | property, Eigenschaft |
| method | method, Funktion |
| DOMString | String |
| unsigned long | int, oder long |
| Bezeichner | bleiben gleich |

Bemerkungen

- Interface Teile von Level 1 Core werden u.U. vom Browser in JavaScript nicht unterstützt
- die Teile von Level 1 HTML werden in der Regel unterstützt

Unterschiede zwischen NS 4.x und IE 5.0

- Objekte die per ID/Name selektiert werden benötigen 2 Zusätze
- Collection aller Properties "all"
- Stilattribut "style"

- `document+"."+all+"."+elementID+"."+style`
- NS `document.elementID` entspricht
IE `document.all.elementID.style`

Unterschiede zu NS 6.x (und IE 5.5 ?)

- Objekte die per ID/Name selektiert werden benötigen einen DOM Funktionsaufruf
- `getElementById('elementID')`
- also `document.getElementById('elementID')`
anstelle von `document.elementID` bzw.
`document.all.elementID.style`

```
var coll = "";
var styleObj = "";
var dhtml = 0;
var dom = 0;

if (document.getElementById) { // w3c
    dom = 1;
} else if (document.layers) { // ns
    dhtml = 1;
} else if (document.all) { // ms
    dhtml = 1;
    coll = "all."; styleObj = ".style";
}
```

```
function getObj(obj){
    if ( (dom==1) && (typeof obj == "string") ){
        var xobj = document.getElementById(obj);
        if (xobj.style) xobj = xobj.style;
        return xobj; }
    if ( (dhtml==1) && (typeof obj == "string") ){
        var xobj = eval ("document."+coll+obj+styleObj);
        return xobj; }
    else { return obj; }
}
```

```
function show_propsobj(obj, obj_name) {
    var result = "";
    var xobj = getObj(obj);
    for (var i in xobj) {
        result += obj_name + "." + i + " = " + xobj[i] + "\n";
    }
    return result;
}
```

```
<form name="anzeigeobj" action="none" >
    Eingabe: <code>document.</code>
    <input type="text" name="eingabe" value="" size="40" />
    <input type="button" value="show_propsobj(eingabe)"
        onclick="alert(show_propsobj(eval(document.forms[1].elements[0].value), 'document.'+document.forms[1].elements[0].value))" />
</form>
```

Eingabe: `document`. **Textfield:** `eingabe` **Button:** `show_propsobj(eingabe)`

Beispiele:

```
forms[0]
forms[1].elements[0]
anzeige
anzeige.style
```

Ausgabe mit `document.write`.

12.5. Ausblick

- DOM Level 2, Version 1.0, W3C Recommendation vom 13. November 2000
- mit Ausnahme des HTML Teils, W3C Recommendation, 9. Januar 2003.

DOM Level 2

- aufgeteilt in 14 Module
- Zugriff auf Stilinformation
- Zugriff auf Events (Ereignisse)
- Navigation im Dokumenten-Baum
- Validierung entsprechend DTD

Module

- Core Modul
- XML Modul
- HTML Modul
- Views Modul
- Style Sheets Modul
- CSS1 Modul
- CSS2 Modul
- Events Modul
- User-Interface-Events Modul
- Mouse Events Modul
- Mutation Events Modul
- HTML Events Modul
- Range Modul
- Traversal Modul

Änderungen gegenüber Level 1

- zusätzliche Interfaces wegen Namespaces
- bei Node: `namespaceURI`, `prefix`, `localName`
- bei Document: `createElementNS`, `getElementsByNameNS`, `createAttributeNS`
- bei Element: `xxxAttributeNS` wobei `xxx` = `set`, `get`, `remove`; `xxxAttributeNodeNS`

Stilvorlagen allgemein

- neue Interfaces
- `StyleSheet`

- StyleSheetList
- MediaList
- DocumentStyle
- LinkStyle

CSS Stilvorlagen

- ca. 34 neue Interfaces, u.a.
- StyleSheet
- CSSStyleSheet
- CSSMediaRule
- CSS2Counter...
- CSS2FontFaceSrc
- RGBColor
- CSS2Properties

Ereignisse

- 6 neue Interfaces
- Event
- EventListener, EventTarget
- MutationEvent, UIEvent
- MouseEvent
- KeyEvent gibts in Recommendation nicht mehr

Weitere Interfaces

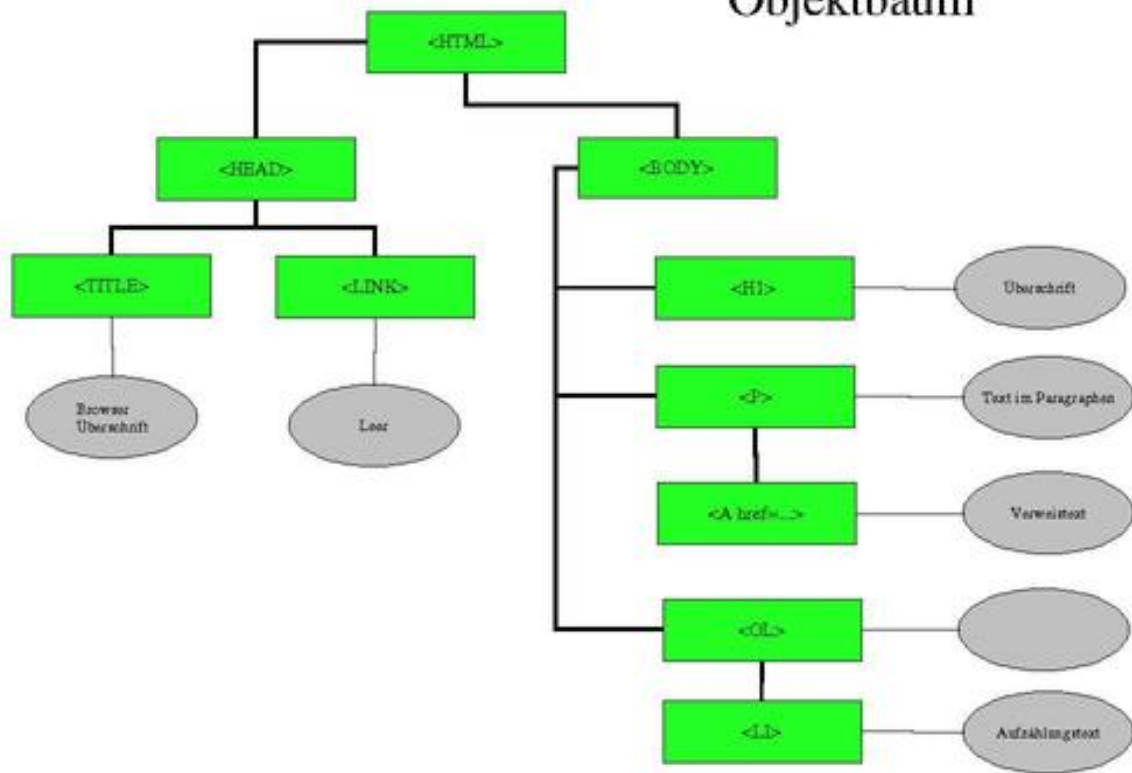
- NodeFilter
- NodeIterator
- TreeWalker, DocumentTraversal
- wegen XML: Range, RangeException

DOM und XML

- wichtig zum Verständnis von XLL
Extended Linking Language, XLink, XPointer
- wichtig zum Verständnis von XSL
Extensible Stylesheet Language

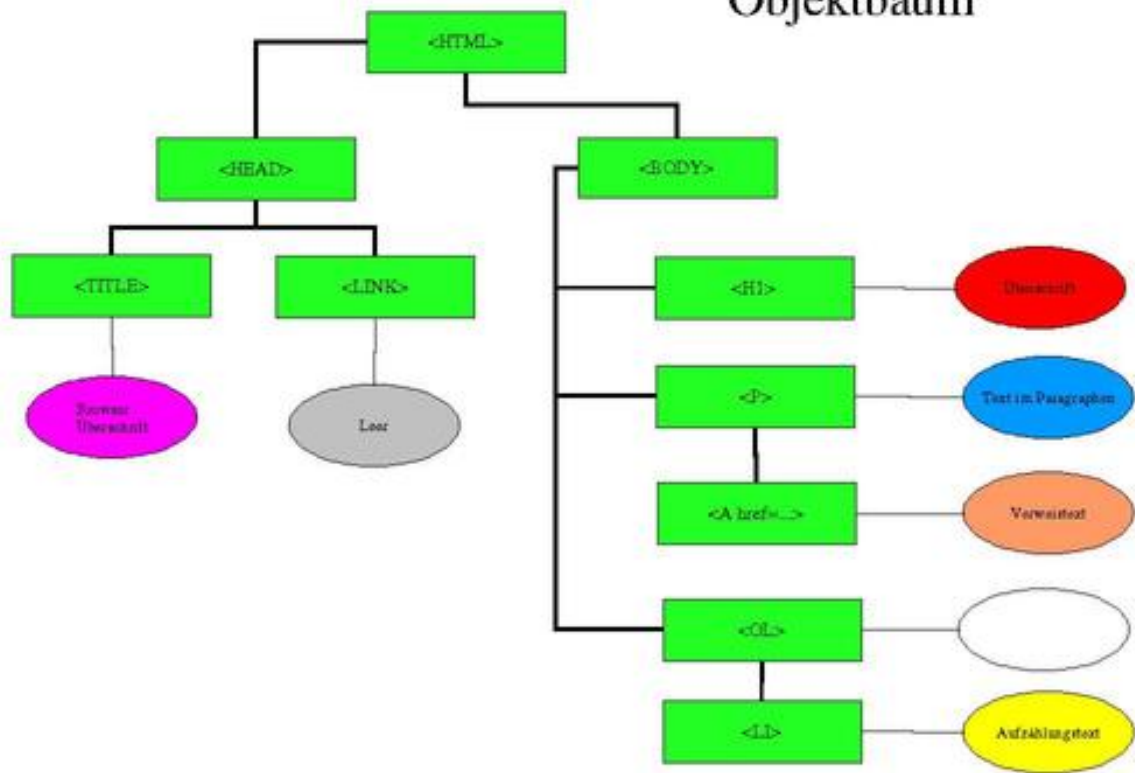
12.6. Zusammenfassung

Objektbaum



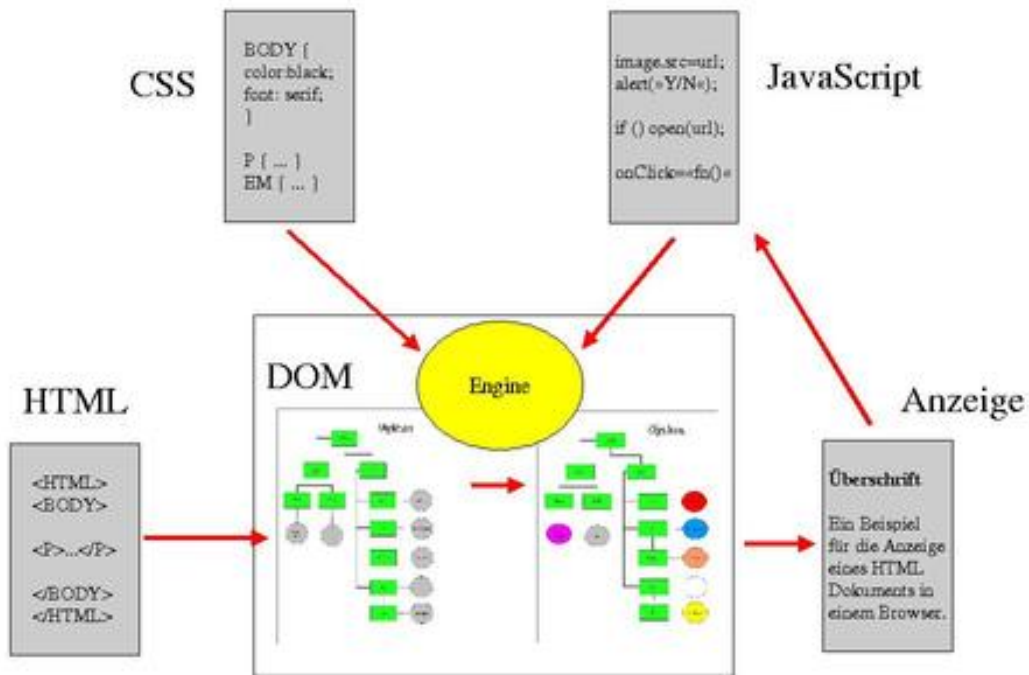
Objektbaum nach dem Parsen

Objektbaum



Objektbaum nach Modifikation durch CSS

HTML, CSS, JavaScript und DOM



Zusammenspiel der Einzelteile

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Wed May 26 23:40:34 CEST 2004

13. Namespaces und XLink

- XML Helfer im Detail
- Namensräume
- XML Infoset
- XLink

13.1. Namensräume



Erleichtert die Verwendung verschiedener DTDs im gleichen Dokument.

- Vordefinierte Namensräume
xml immer auf w3c/XML/namespace
und xmlns immer leer
- Default Namespace xmlns="" ...
Definierte Namespaces xmlns:spec="" ...
- Definition von spec, xsl und html:

```
<X xmlns:spec="http://www.w3.org/specpath/" >  
  <spec:tag ...> ... </spec:tag>  
</X>
```

```
<xml:stylesheet  
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
  xmlns:html="http://www.w3.org/TR/REC-html40/" >
```

- Verwendung

```
<em> ... </em>  
<latex:em> ... </latex:em>  
<HTML:A HREF="...">Beschreibung</HTML:A>  
<person HTML:href="..."> ... </person>
```

13.2. XML Infoset

XML Infoset definiert die wichtigsten Begriffe (information items) und deren Spezifikation und Bedeutung in der XML Welt.

Document Item

hat Kind-Elemente, Version, Character Encoding, unparsed Entities

Element Item

hat Namensraum, Kind-Elemente, Attribute, Eltern-Element

Attribute Item

hat Namensraum, Datentyp, References, Owner-Element

Processing Instruction Item

hat Target-Name, Inhalt, Eltern-Element

Unexpanded Entity Reference Item

hat Name, System oder Public Identifier, Eltern-Element

Character Item

hat Character-Code nach ISO/IEC 10545, Universal Multiple-Octet Coded Character Set (UCS), Whitespace Behandlung, Eltern-Element

Comment Item

hat Inhalt und Eltern-Element

Document Type Declaration Item

hat System oder Public Identifier, Kind-Elemente und Eltern-Element

Unparsed Entity Item

hat Name, System oder Public Identifier, Eltern-Element

Notation Item

hat Name, System oder Public Identifier

Namespace Item

Hat Prefix und Namen des Namensraums

Andere Spezifikationen definieren die syntaktischen und strukturellen Erscheinungsformen dieser Items.

13.3. XLink



Am Anfang XLL, jetzt aufgeteilt in XLink und XPointer.

Verweise zwischen mehreren 'Ressourcen'. Metadaten für Verweise. Eignung für Link-Datenbanken.

W3C Recommendation seit Juni 2001.

- Auswahl des Namensraums für XLink

```
xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
```

- Definition durch Attribut

```
<A xlink:type="simple" ... >
```

oder Element (mit Recommendation nicht mehr Normativ)

```
<xlink:simple href="..." ... > Inhalt </xlink:simple>
```

- mögliche Typen sind
 - simple wie A in HTML
 - extended erweitert, volles XLink
 - locator nur externer Verweis
 - arc Mit Hinweisen über die Richtung von Links
 - resource lokaler Verweis

- title nur zur Beschreibung

- Locator Attribut, Verweis

```
xlink:href="connector"
```

- Verbinder, Connectors
- URI#XPointer Client kümmert sich um Auflösung
- URI|Xpointer Server kümmert sich um Auflösung
- URI?CGI-Parameter
- Anwendung

```
<L xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
  xlink:type="simple"
  xlink:href="http://host/path/doc.html#xyz"
> text </L>
```

Verhaltensattribute von Links

- xlink:show="..." Anzeigeverhalten
 - new Anzeige in neuem Fenster
 - replace Anzeige im aktuellen Fenster
 - embed Einfügen in das aktuelle Fenster
 - other Verhalten evtl. anderweitig definiert
 - none Verhalten nicht definiert
- xlink:actuate="..." Aktivierungsverhalten
 - onLoad automatische Anzeige
 - onRequest Anzeige nur nach Benutzerwunsch
 - other Verhalten evtl. anderweitig definiert
 - none Verhalten nicht definiert
- xlink:label="NMTOKEN"
 - xlink:from="NMTOKEN"
 - xlink:to="NMTOKEN"
 - Beschreibung der Verlinkung bei arc
 - Verwendung zusammen mit xlink:role
- xlink:role="URI" freie Zusatzinformationen, maschinenverwendbar
- xlink:arcrole="URI" freie Zusatzinformationen, maschinenverwendbar
- xlink:title="CDATA" freie Zusatzinformationen für Menschen verwendbar

Gültige Kombinationen von Attributen

| Quelle W3C, 1999, 2001 | | | | | | |
|------------------------|--------|----------|---------|-----|----------|-------|
| | simple | extended | locator | arc | resource | title |
| type | R | R | R | R | R | R |
| href | O | | R | | | |
| role | O | O | O | | O | |

Namespaces und XLink

| | | | | | | |
|---------|---|---|---|---|---|--|
| arcrole | O | | | O | | |
| title | O | O | O | | O | |
| show | O | | | O | | |
| actuate | O | O | | O | | |
| label | | | O | | O | |
| from | | | | O | | |
| to | | | | O | | |

R = required, O = optional

Beispiel

```
<X xlink:type="extended" >
<L xlink:type="locator"
  xlink:role="TR" xlink:title="Übersetzung"
  xlink:show="new" xlink:href="/cgi-bin/xlate?term=Verweis" />
<L xlink:type="locator"
  xlink:role="Kontext" xlink:title="Links im Kontext"
  xlink:show="replace" xlink:href="link-spec.html#verweis" />
<L xlink:type="locator"
  xlink:role="Bild" xlink:title="Links in Bildern"
  xlink:show="embed" xlink:href="bild.gif" />
<L xlink:type="locator"
  xlink:role="Tutorium" xlink:title="Link Tutorium"
  xlink:show="new" href="xml-tut.html#ID(def-link)..DITTO,next(3,P)" />
Verweise
</X>
```

mit der DTD

```
<!ELEMENT X (#PCDATA|L)* >
<!ELEMENT L EMPTY >

<!ATTLIST X xlink:type CDATA #FIXED "extended" >
<!ATTLIST L xlink:type CDATA #FIXED "locator" >
```

erzeugt (abhängig vom UA) u.U. folgendes Menue

```
- Übersetzung
- Links im Kontext
- Links in Bildern
- Link Tutorium
```

Beispiel für HTML Anchors

```
<!ELEMENT A (#PCDATA) >
<!ATTLIST A xmlns:xlink="http://www.w3.org/1999/xlink/namespace/" >
<!ATTLIST A xlink:type "simple" >
<!ATTLIST A xlink:href CDATA #REQUIRED >
<!ATTLIST A xlink:show "replace" >
<!ATTLIST A xlink:actuate "onRequest" >
```

XLink in Mozilla / Netscape 6

Mozilla unterstützt einfache Xlinks.

- **type:** nur simple Links, keine extended Links
- **show:** new und replace, kein embedd

- **actuate**: teilweise onLoad
- **href**: wie in HTML

Xlink [Beispiel](#)

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat Jun 12 12:41:19 CEST 2004

14. XPath und XPointer

- XPath
- XPointer

14.1. XPath



Verweise auf Teile/Parts von XML Dokumenten (als Baum von Knoten betrachtet). Identifikation von Parts durch Vergleich von Zeichenketten.

Wird in XSLT und XPointer zur Spezifikation von Fragmenten verwendet. Diese definieren auch den Kontext in dem die XPath Ausdrücke ausgewertet werden.

Aufbau eines Pfadbestandteils

```
achse::knotentest[prädikat]
```

Beispiel

```
child::para[position()=7]  
http://host/pfad/resource#xpointer(child::para[position()=7])
```

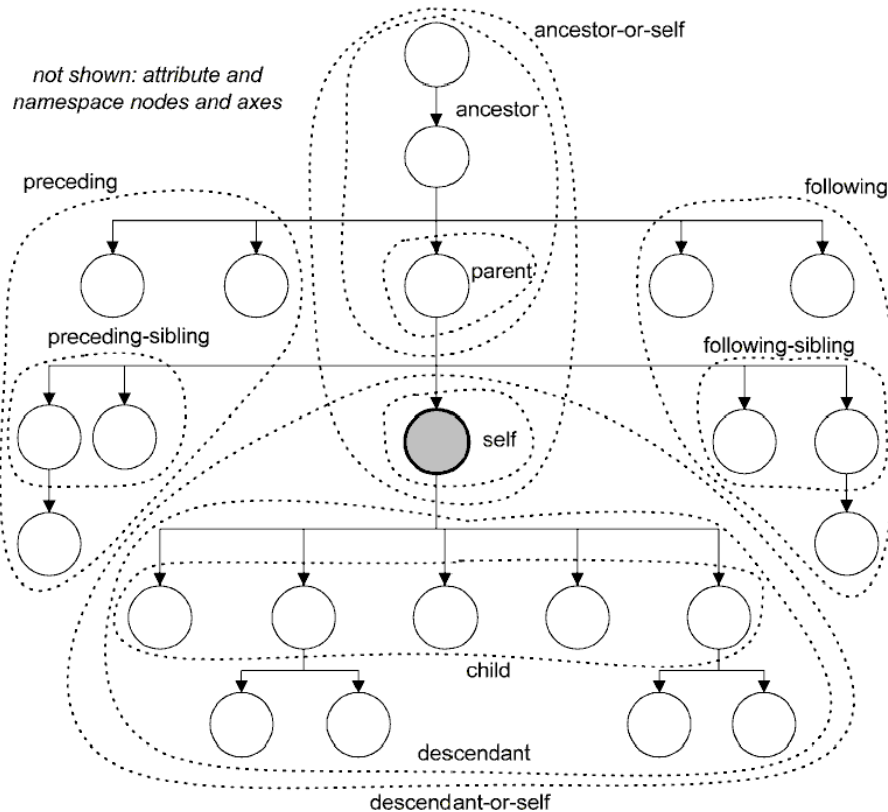
Komposition von Pfaden durch Pfadbestandteile, die durch / getrennt werden.

- Achsen (axis): definieren Folgen von Kandidaten zur Auswahl
- Prädikate (predicates): Boolesche Bedingungen zur Einschränkung der Auswahl
- Funktionen: generieren von (neuen) Kandidaten zur Auswahl

Achsen

- child Kindknoten des aktuellen Knotens
- descendent Nachfahren
- parent Elternknoten
- ancestor Vorfahren
- preceding Vorgänger Knoten
- following Nachfolger Knoten
- preceding-sibling Vorgänger Geschwister-Knoten
- following-sibling Nachfolger Geschwister-Knoten

- attribute Attributknoten
- namespace Namensraum-Knoten
- self der Knoten selbst
- descendant-or-self
- ancestor-or-self



XPath Achsen, Quelle Crane Softwrights, 2001

ancestor, descendant, following, preceding und self Partitionieren die Menge aller Knoten eines Dokuments.

Knoten-Tests

- achse::* zur Selektion aller Knoten
- achse::name zur Selektion bestimmter Elemente oder Attribute
- achse::funktion() zur Selektion von Knoten mit bestimmten Eigenschaften.

Beispiel: alle para Element-Knoten

```
child::para
```

Beispiel: das href Attribut des Knotens

```
attribute::href
```

Beispiel: alle Textknoten

```
child::text()
```

Prädikate

- `[ test-ausdruck ]`
Selektiert Knoten, für die die Auswertung des Test-Ausdrucks "wahr" ergibt
- Ausdrucksbedeutung von `axis::node[test]`
ist `(axis::node)[test]`
und nicht `axis::(node[test])`
- der Test-Ausdruck darf arithmetische (+, -, *, div, mod) und boolesche (or, and, !, <, >, <=, >=, =, !=) Operatoren enthalten.
Schreibweise: `<` für <

Beispiel: der 7. Knoten

```
[ position() = 7 ]
```

Funktionen

- `last()` Position des letzten Elements
- `position()` aktuelle Position des Elements
- `count(node-set)` Anzahl der Elemente
- `id(object)` Element, das durch Objekt Identifiziert wird
- `name(node-set)` Name der Selektion
- `string(object)` Konvertiert das Objekt in Zeichenkette
- `concat(string1, string2 ,...)` Zeichenketten Verbindung
- `starts-with(string1, string2)`
- `contains(string1, string2)`
- `substring(string, position, length)`
- `string-length(string)`
- `translate(string1, string2, string3)`
- `boolean(object)`
- `not(boolean)`
- `number(object)`
- `round(number)`
- ...

Abkürzungen

Zur Vereinfachung gibt es eine ganze Reihe von kompakteren Bezeichnungen der Pfadbestandteile.

- `para` für `child::para`
- `*` für `child::*`
- `text()` für `child::text()`
- `@href` für `attribute::href`
- `@*` für `attribute::*`
- `[ 7 ]` für `*::*[position()=7]`

- `chapter//para` für `child::chapter/descendant::para`
- `chapter/section` für `child::chapter/child::section`
- `//` für `/descendant-or-self::node()/`

14.2. XPointer



Verweise auf Teile/Fragmente von XML Dokumenten. Fragmente sind einzelne Punkte/Objekte/Knoten und zusammenhängende Bereiche von Punkten. Identifikation von Fragmenten durch Vergleich von Zeichenketten.

W3C Recommendation seit März 2003 in drei Teilen:

- XPointer Framework
- XPointer `element()` Scheme
- XPointer `xmlns()` Scheme
- XPointer `xpointer()` Scheme ist noch Working Draft

Steht in enger Beziehung zu DOM. Baut auf XPath auf. Wird zusammen mit XLink zur Definition von XML Links benötigt.

URL Aufbau mit XPointer

```
service:://host/pfad/resource#xpointer-expr  
service:://host/pfad/resource|xpointer-expr
```

dabei ist `xpointer-expr`:

```
html-name  
oder xpointer( xpath-expr )  
oder xpointer( xpath-expr to xpath-expr )  
oder element( elem-expr )  
oder xmlns(ns=url)
```

Beispiele

```
#intro  
#xpointer(id("intro"))  
#xpointer(//*[@id="intro"])  
#xmlns(h=urlh)xmlns(x=urlx)h:elem/x:elem  
#element(append/2/1)
```

Erweiterungen gegenüber XPath

neue Typen `point` (ein Knoten oder ein Zeichen einer Zeichenkette) und `range` (`begin-expr to end-expr`) und einige dazu gehörige Funktionen, z.B. `point()` und `range(...)`.

Beispiele

```
xpointer(  
  id("sec2.1")/descendant::P[last()] to  
  id("sec2.2")/descendant::P[first()]  
)
```

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:38:26 CEST 2004

15. XML Style Sheets (XSL)

- Einleitung
 - XSL Transformations
 - Beispiele und Tools
-

15.1. Einleitung



Am Anfang XSL, jetzt aufgeteilt in XSL Transformations (XSLT) und XSL Formatting (FO).

- wesentlich mächtiger als CSS, z.B. Inhaltsverzeichnisse
- DSSSL in XML Syntax
- Document Type ist `xsl:stylesheet`
- kann zusammen mit CSS verwendet werden
- Formatting Objects (FO) bezeichnen die elementaren formatierbaren Objekte bei DSSSL flow objects
- XSL-Formatting Spezifikation definiert FO Vokabular
- Bearbeitung erzeugt Baum von FO
- XSL - Transformations Spezifikation definiert Transformationskonstrukte
- `xsl:template` wählt Element aus und beschreibt seine Formatierung
- `xsl:apply-templates` stößt Bearbeitung des Inhalts an

Varianten und Beispiele

CSS:

```
em { font-family: roman }
```

CSS mit XML:

```
<?xml-stylesheet href="url" type="text/css"
  media="screen" ... ?>
```

XSLT to HTML:

```
<xsl:template match="em">
  <I>
    <xsl:apply-templates/>
  </I>
</xsl:template>
```

(XSLT to CSS:)

```
<xsl:template match="em">
  <css:chunk font-family="roman">
    <xsl:apply-templates/>
  </css:chunk>
</xsl:template>
```

XSLT to LaTeX:

```
<xsl:template match="em">
  <xsl:text>\em </xsl:text>
  <xsl:apply-templates/>
  <xsl:text></xsl:text>
</xsl:template>
```

Beispiel eines XML Dokuments

- [Report DTD](#)
- [Dokument](#)

Konversion nach HTML und LaTeX mit Xalan/Xerces

- [Text Style, Ergebnis](#)
- [HTML Style, Ergebnis](#)
- [LaTeX Style, Ergebnis, PostScript, PDF](#)

15.2. XSL Transformations Sprachelemente

Struktur und Aufbau von XSLT Style Sheets

- `xsl:import` Einfügen von externen Definitionen (überschreibend)
- `xsl:include` Einfügen von externen Definitionen (kombinierend)
- `xsl:id` Definition von eindeutigen Bezeichnern
- `xsl:strip-space elements="..."`, `xsl:preserve-space elements="..."` Entfernen oder Einfügen von Leerraum im Inhalt für bestimmter Elemente
- `xsl:variable`, `xsl:param` Definition von Variablen und Parametern
- `xsl:key` Definition von Schlüsseln
- `xsl:decimal-format`, `xsl:namespace-alias` Definition von Zahlenformaten und Aliasen
- `xsl:attribute-set` definiert Liste von Attributen
- `xsl:output` Definition der Charakteristika der Ausgabe
- `xsl:template` Definition von Ersetzungsmustern
 `<xsl:template match="pattern">` Definition eines Erkennungsmusters
- alles in (fast) beliebiger Reihenfolge im Top-Level

Aus Knoten (nodes) eines Dokuments werden Ergebnisknoten gebildet.

Knoten sind: Elemente, Attribute, etc.

Patterns, Muster in match und select

- jetzt Spezifikation in XPath mit Erweiterungen durch XSLT eigene Funktionen
- Erkennung erfolgt relativ zum aktuellen (current) Knoten

- p1 | p2 mehrere Patterns als Alternativen
- n1/n2 Kontextangabe, n2 als Subknoten von n1, eine Ebene
- / Root Knoten
- n1//n2 Kontextangabe, n2 als Subknoten von n1, beliebig tiefere Ebene
- */p Wildcard, p als Subknoten irgendwas
- . aktueller (current) Knoten
- .. Elternknoten
- @attname selektiert ein bestimmtes Attribut
- [test] selektiert Knoten, für die "test" zu trifft

`list[@type="ordered"]`

selektiert Listen vom Typ "geordnet"

- weitere Tests:
first-of-any(), last-of-any()
first-of-type(), last-of-type()
id(n), ancestor()
- weitere Funktionen:
document(obj, ns), key(string, obj), format-number(num, string, string?),
- Beispiel

`section/list[@type="ordered" and last-of-type()]`

Erzeugen von Knoten in Templates

- Templates erzeugen Stil-Ausgabe und stossen die Verarbeitung von (Sub-) Knoten an.
- Literale, die XSL nicht kennt, werden in den (Ausgabe-) Knoten kopiert
- `xsl:text` oder unbekannte Literale erzeugen einen Text-Knoten
- `xsl:element name="..."` erzeugt einen extra Element-Knoten mit angegebenem Namen
- `xsl:attribute name="..."` erzeugt einen extra Attribut-Knoten mit angegebenem Namen
- `xsl:processing-instruction` erzeugt einen Processing-Instruktion-Knoten

```
<xsl:processing-instruction name="php">echo "Hallo"; </xsl:pi>
erzeugt
```

```
<?php echo "Hallo"; ?>
```

- `xsl:comment` erzeugt einen Kommentar-Knoten
- `xsl:message terminate="yes"|"no"` erzeugt eine Nachricht und wartet ggf. auf Eingabe vom Anwender

Verarbeitung in Templates

- `xsl:apply-templates select="pattern"` allgemeiner Aufruf der Template-Verarbeitung
- `select="pattern"` ist (bei allen Verarbeitungstemplates) optional, falls vorhanden werden nur Knoten des pattern-Typs ausgewählt, sonst alle
- `xsl:for-each select="..."` Verarbeitung für alle (angegebenen) Subknoten
- `mode="bezeichnung"` wählt nur Templates mit dem gleichen mode-Attribut
- `xsl:sort order=(ascending|descending)`
`data-type=(text|number)` `lang=language`
`case-order=(upper-first|lower-first)`
Sortiert entsprechend den Kriterien in `xsl:apply-templates` und `xsl:for-each`

- `xsl:number level=(single|multi|any) count="bezeichner" from="start-wert" format="beispiel"`
erzeugt Nummern-Knoten entsprechend den Vorgaben
- `format="beispiel"` erzeugt Nummernknoten entsprechend dem Format-Beispiel
`beispiel=(1|A|a|i|I|001)`
`digit-group-sep=","`
`n-digits-per-group="3"`
- Einfügen von Werten in Ausgabetexte mit `{ }`
z.B. `src="{@attribut}"` oder `src="{ $variable}"`
- `xsl:if test="bedingung"` Verarbeitung nur falls die Bedingung wahr ist
- `xsl:choose` mit
`xsl:when test="bedingung"` Verarbeitung wie in "switch" nur falls die Bedingung wahr ist
- `xsl:copy` Kopiert den aktuellen Knoten
- `xsl:value-of select="pattern"`
berechnet den Wert des angegebenen Musters, z.B. eines Attributs
- `xsl:constant name=".." value=".."`
definiert (Literal-) Konstanten

15.3. Beispiele und Tools

Beispiele aus dem [Camena Projekt](#).

Tools zur Verwendung von XSLT

Xalan XSLT Prozessor, von der Apache Gruppe.

transform:

```
#!/bin/sh
# echo "CLASSPATH:" $CLASSPATH
VALIDPATH="/home/kredel/java/lib/xalan.jar:/home/kredel/java/lib/xalansamples.jar:/home/kredel/java/lib/xerces.jar"
export CLASSPATH=$VALIDPATH:$CLASSPATH
# echo "CLASSPATH:" $CLASSPATH
if [ $# -eq 2 ]
then
/usr/lib/jdk1.3/bin/java org.apache.xalan.xslt.Process -DIAG -in $1 -xsl $2
elif [ $# -eq 3 ]
then
/usr/lib/jdk1.3/bin/java org.apache.xalan.xslt.Process -DIAG -in $1 -xsl $2 -out $3
else
/usr/lib/jdk1.3/bin/java org.apache.xalan.xslt.Process $*
fi
```

transform.bat:

```
set VALIDPATH=u:\xerces\xalan.jar;u:\xerces\xalansamples.jar;u:\xerces\xerces.jar
set CLASSPATH=%VALIDPATH%;%CLASSPATH%
echo "CLASSPATH:" %CLASSPATH%
java org.apache.xalan.xslt.Process -in %1 -xsl %2 -out %3
```

xalan.xslt.Process Optionen:

```
> transform
Optionen der Klasse Process in Xalan-J-Befehlszeile:
  -IN inputXMLURL
  [-XSL XSLTransformationURL]
```

```
[-OUT outputFileName]
[-E (Entity-Referenzen nicht erweitern)]
[-QC (Geräuscharme Warnungen bei Musterkonflikten)]
[-TT (Vorlagen beim Aufruf verfolgen.))
[-TG (Jedes Erzeugungsereignis verfolgen.))
[-TS (Jedes Auswahlereignis verfolgen.))
[-TTC (Die Vorlagen-Tochterknoten bei Bearbeitung verfolgen.))
[-TCLASS (TraceListener-Klasse für Trace-Erweiterungen.))
[-EDUMP {optionaler Dateiname} (Speicherauszug bei Fehler.))
[-XML (XML-Formatierer verwenden und XML-Header hinzufügen.))
[-TEXT (Einfachen Textformatierer verwenden.))
[-HTML (HTML-Formatierer verwenden.))
[-PARAM Namensausdruck (Stylesheet-Parameter festlegen)]
[-L Zeilennummern für Quelldokument verwenden]
[-MEDIA mediaType (use media attribute to find stylesheet
associated with a document.))
[-FLAVOR flavorName (Explicitly use s2s=SAX or d2d=DOM
to do transform.))
[-DIAG (Print overall milliseconds transform took.))
[-URIRESOLVER vollständiger Klassenname (zum Auflösen von
URIs zu verwendender URIResolver)]
[-ENTITYRESOLVER vollständiger Klassenname (zum Auflösen von
Entities zu verwendender EntityResolver)]
[-CONTENTHANDLER vollständiger Klassenname (zum Serialisieren
der Ausgabe zu verwendender ContentHandler)]
```

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sun May 23 12:53:24 CEST 2004

16. XSL Formatting Objects

- Einleitung
 - XSL-FO Sprachelemente
 - XHTML nach FO nach PDF
-

16.1. Einleitung

Darstellung von Online- und Druck-Dokumenten sowie Sprachausgabe.

Spezifikation der grundlegenden Elemente, ohne die Details einer Implementierung festzulegen.

für Seiten

Breite, Höhe, Ränder, Bereiche, Regionen

für Blöcke und Inline-Elemente

Breite, Höhe, Ränder

für Blöcke

Absätze, Listen, Tabellen, Bilder

für Inline-Elemente

Links (a), Font-Attribute

für Grafiken

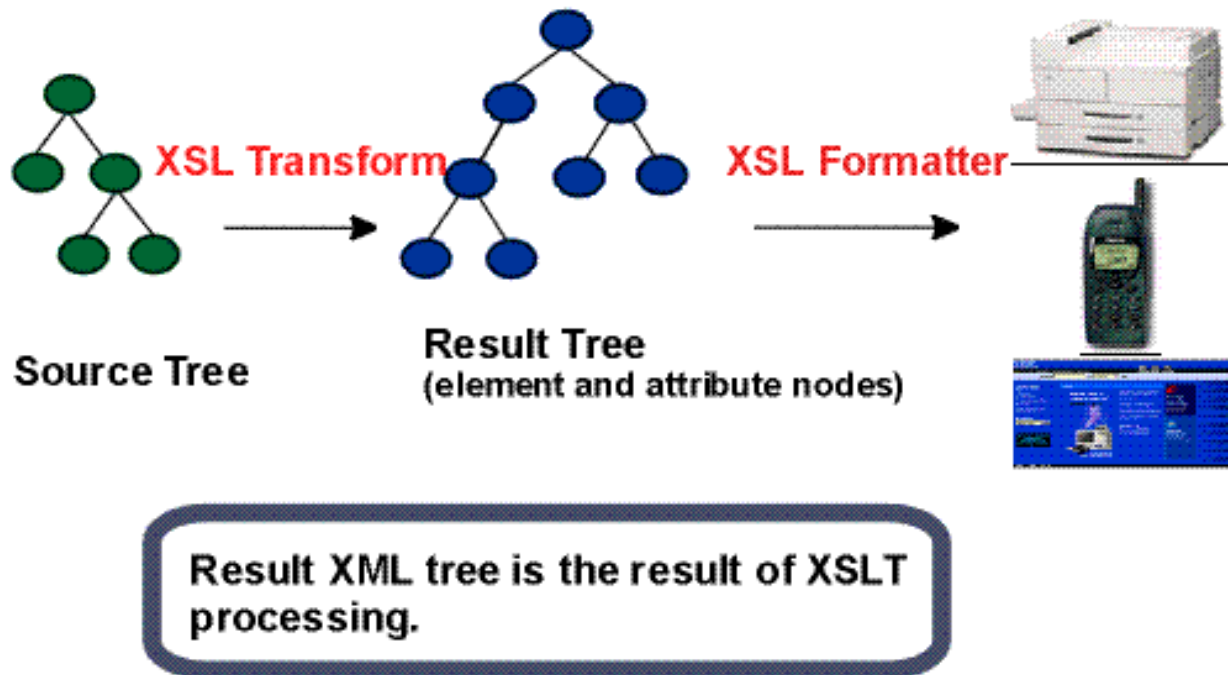
Einbindung verschiedener Formate

für Sprachausgabe

wie in CSS2

für Internationalisierung

Unicode und Textschreibrichtungen: lr-tb, rl-tb und tb-rl.



Der XSL-FO Verarbeitungsprozess (Quelle W3C)

Der Verarbeitungsprozess verläuft intern durch verschiedene Verfeinerungen von fo:-Bäumen über object/property-Bäumen zu object/traits-Bäumen. Die Letzteren können dann direkt auf dem Ausgabemedium dargestellt werden.

3 Dinge bringt XSL-FO unter einen Hut:

Spezifikation der Darstellungselemente

fo:block, fo:inline mit Eigenschaften (properties), die CSS2 entsprechen.

Spezifikation der Charakteristika des Ausgabemediums

für die Druckausgabe (PDF) fo:layout-master-set, fo:simple-page-master fo:region-body oder ähnliche Dinge
für Bildschirmausgabe oder Sprachausgabe

Beschreibung des (Inhalts)Flusses

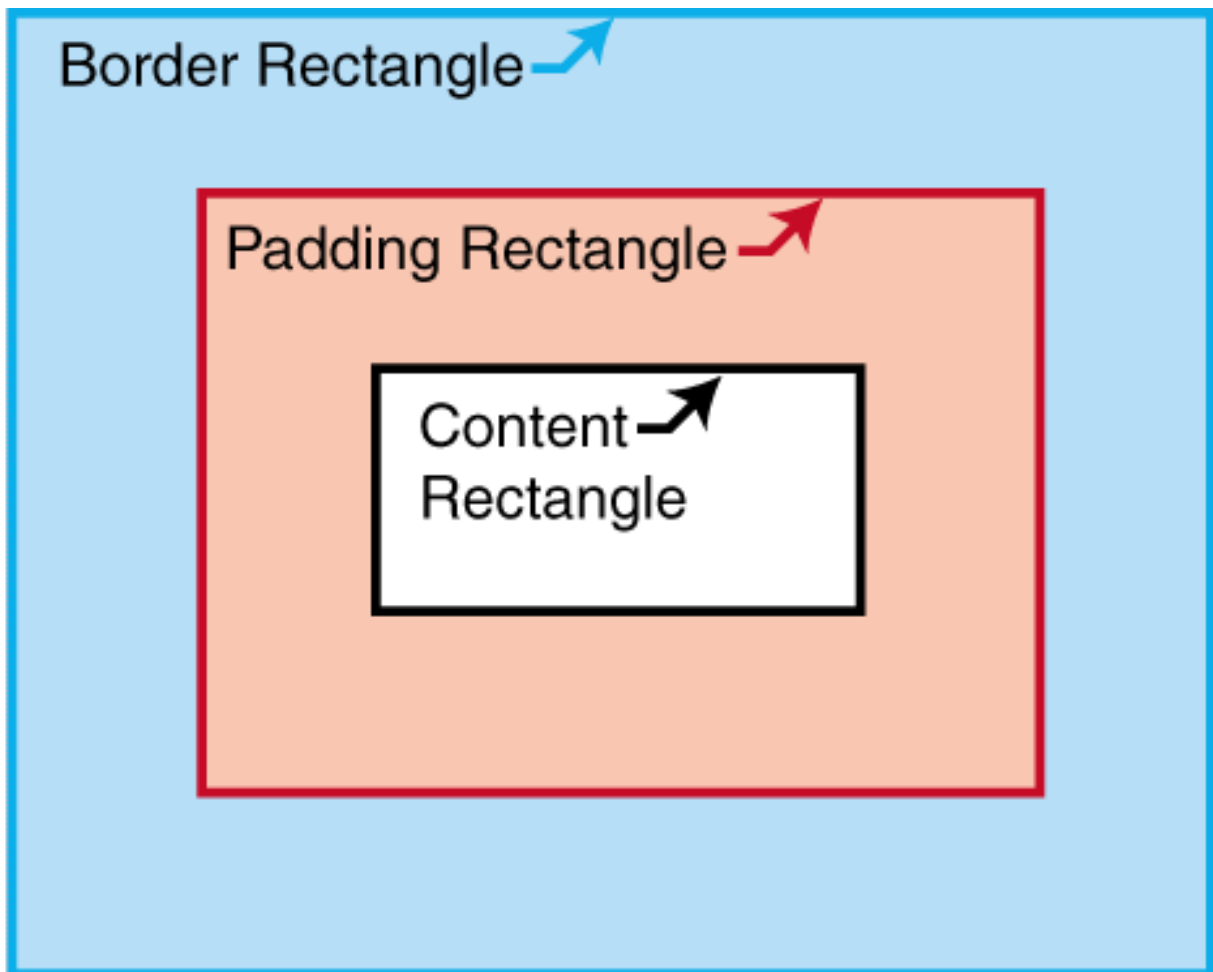
wie "fließen" die Inhaltselemente in das Ausgabemedium: fo:page-sequence master-reference="seite",
fo:static-content, fo:flow flow-name="body"

16.2. XSL Formatting Objects Sprachelemente

Als Namensraum wird in diesem Abschnitt immer fo: mit dem URL <http://www.w3.org/1999/XSL/Format> verwendet. Das Root-Element ist immer fo:root.

elementare Darstellungen:

Formatierung durch *areas*, die eine Erweiterung des Box-Modells von CSS (block oder inline) sind. U.A. können auch Zwischenräume *spaces* genauer definiert werden sowie Nebenbedingungen (constraints) für die Anordnung der Areas.



XSL-FO Area Rechtecke (Quelle W3C)

- `fo:block` Formatierung eines Blocks mit diversen Attributen
- `fo:character` Formatierung eines Zeichens mit diversen Attributen
- `fo:inline` Formatierung einer Zeichenfolge mit diversen Attributen innerhalb eines `fo:block`
- `fo:basic-link` Formatierung eines A-Links mit internen und externen Referenzen
- `fo:external-graphic` Formatierung einer Grafik (z.B. GIF-Image) mit diversen Attributen
- `fo:list-block` und `fo:list-item` für die Formatierung einer Liste von Blocks und Listen-Items mit diversen Attributen
- Formatierung von Tabellen mit diversen Attributen `fo:table`, `fo:table-row`, `fo:table-cell`, `fo:table-column`, `fo:table-body`, ...

Attribute/Properties angelehnt an CSS2

Die CSS2 Attribute sind zum Teil weiter ausdifferenziert.

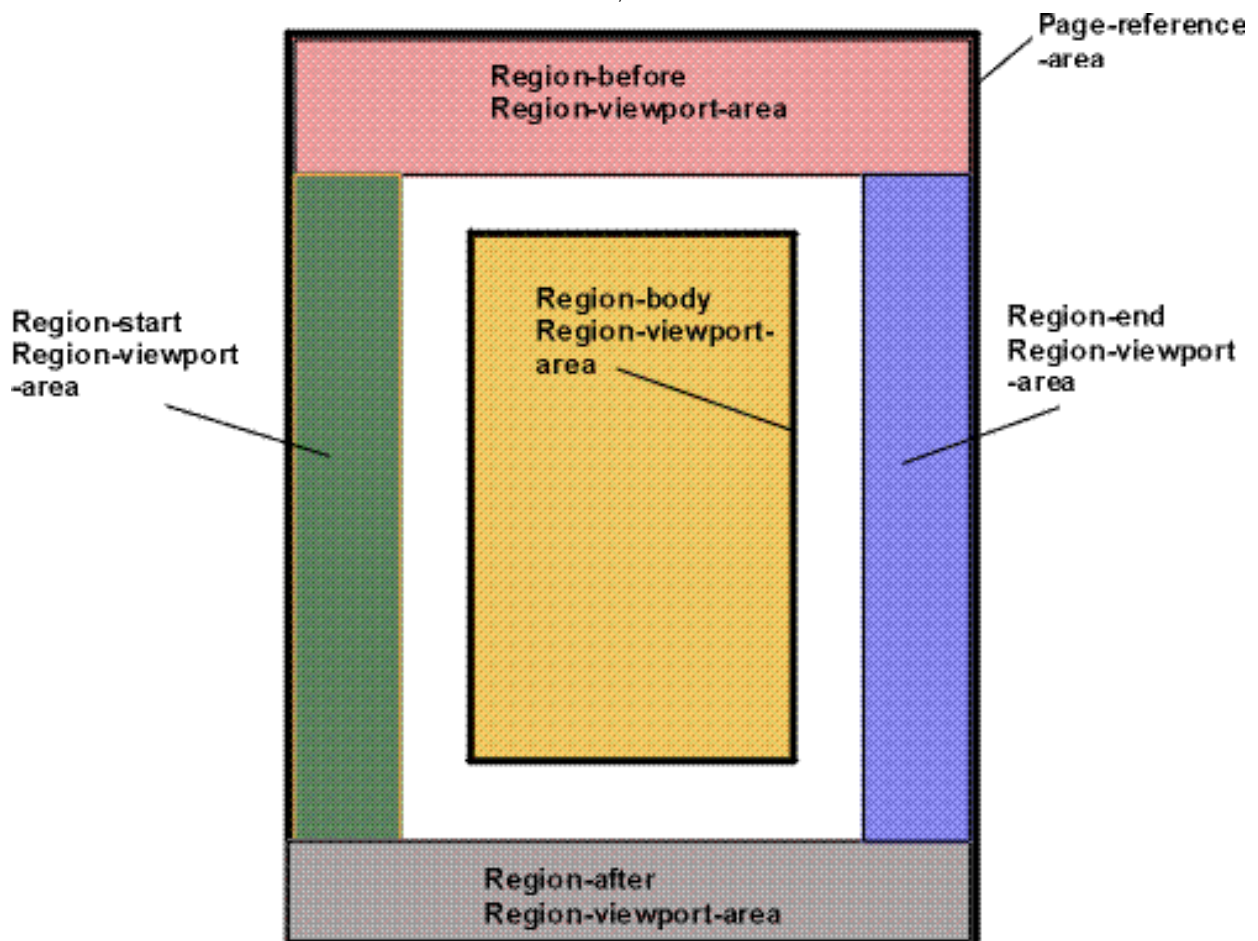
- `font-family="sans-serif"` Attribut für Schriftart
- `font-style="italic"` Attribut für Schriftstil
- `color="red"` Attribut für Schriftfarbe

- `background-color="aqua"` Attribut für Hintergrundfarbe
- `text-align="center"` Attribut für Textausrichtung
- ... und neue Attribute, wie
- `space-before="8pt" space-after="4pt"` Attribute für Leerraum vor bzw. hinter einem Block
- `break-before="page"` Attribut, das einen Seitenumbruch anstösst
- `external-destination="url(' {@href} ')"` Verweis auf einen externen URL (in `fo:basic-link`)
- `src="url(' {@src} ')"` Verweis auf eine externe Grafik (in `fo:external-graphic`)
- `column-width="5cm"` Breite einer Tabellenspalte (in `fo:table-column`). Zur Zeit gibt es in FOP (0.20.5) noch keine automatische Tabellengrößenbestimmung.

Definition der Charakteristika des Ausgabemediums

Hier nur für Druck/PDF-Ausgabe betrachtet.

Die Definitionen umfassen im Wesentlichen die Breite und Höhe, sowie die verschiedenen Ränder.



XSL-FO Seitenbereiche (Quelle W3C)

- `fo:layout-master-set` definiert alle Seitenvorlagen, die in dem Dokument verwendet werden.
- `fo:simple-page-master` definiert eine einfache Seitenvorlage, die alles wesentliche für eine Druckseite und die Regionen einer Druckseite definieren kann.

- `fo:region-body` `region-name="koerper"`, `fo:region-before`, `fo:region-after`, `fo:region-start`, `fo:region-end` definiert die Regionen, die in einer Druckseite auftreten können. (Vergleichbar mit den Frames in HTML)

Beschreibung der Füllung (flow)

Wie werden die Darstellungselemente (z.B. `fo:block`) in die Ausgaberegionen eingefüllt, bzw. wie fließen diese in die Ausgaberegionen?

- `fo:page-sequence` `master-reference="seite"` definiert eine Folge von Seiten, die ein bestimmtes Masterlayout (z.B. `fo:simple-page-master`) verwenden und die Füllung der Seitenregionen.
- `fo:static-content` `flow-name="bottom"` definiert statischen Inhalt, also Inhalt der sich auf jeder Seite wiederholt, für eine bestimmte Ausgaberegion (`flow-name`)
- `fo:flow` `flow-name="body"` definiert dynamischen Inhalt, also den veränderlichen Inhalt der Seiten, für eine bestimmte Ausgaberegion (`flow-name`)

Einfaches Beispiel

von den Apache FOP Beispielen:

```
<?xml version="1.0" encoding="utf-8"?>

<!-- example for a simple fo file. At the beginning the page layout is set.
Below fo:root there is always
- a single fo:layout-master-set which defines one or more page layouts
- an optional fo:declarations
- and a sequence of one or more fo:page-sequences containing the text and formatting instructions
-->

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set>
    <!-- fo:layout-master-set defines in its children the page layout:
    the pagination and layout specifications
    - page-masters: have the role of describing the
      intended subdivisions of a page and the geometry
      of these subdivisions
    In this case there is only a simple-page-master which
    defines the layout for all pages of the text
    -->
    <!-- layout information -->
    <fo:simple-page-master master-name="simple"
      page-height="29.7cm"
      page-width="21cm"
      margin-top="1cm"
      margin-bottom="2cm"
      margin-left="2.5cm"
      margin-right="2.5cm">
      <fo:region-body margin-top="3cm" margin-bottom="1.5cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <!-- end: defines page layout -->

  <!-- start page-sequence
  here comes the text (contained in flow objects)
  the page-sequence can contain different fo:flows
  the attribute value of master-name refers to the page layout
  which is to be used to layout the text contained in this
  page-sequence-->
  <fo:page-sequence master-reference="simple">

    <!-- start fo:flow
    each flow is targeted
    at one (and only one) of the following:
```

```
    xsl-region-body (usually: normal text)
    xsl-region-before (usually: header)
    xsl-region-after (usually: footer)
    xsl-region-start (usually: left margin)
    xsl-region-end (usually: right margin)
    ['usually' applies here to languages with left-right
     and top-down writing direction like English]
    in this case there is only one target: xsl-region-body
-->
<fo:flow flow-name="xsl-region-body">

  <!-- each paragraph is encapsulated in a block element
  the attributes of the block define
  font-family and size, line-height etc. -->

  <!-- this defines a title -->
  <fo:block font-size="18pt"
    font-family="sans-serif"
    line-height="24pt"
    space-after.optimum="15pt"
    background-color="blue"
    color="white"
    text-align="center"
    padding-top="3pt">
    Extensible Markup Language (XML) 1.0
  </fo:block>

  <!-- this defines normal text -->
  <fo:block font-size="12pt"
    font-family="sans-serif"
    line-height="15pt"
    space-after.optimum="3pt"
    text-align="justify">
    The Extensible Markup Language (XML) is a subset of
    SGML that is completely described in this document.
    Its goal is to enable generic SGML to be served,
    received, and processed on the Web in the way
    that is now possible with HTML. XML has been designed
    for ease of implementation and for interoperability
    with both SGML and HTML.
  </fo:block>
  ...
</fo:flow> <!-- closes the flow element-->
</fo:page-sequence> <!-- closes the page-sequence -->
</fo:root>
```

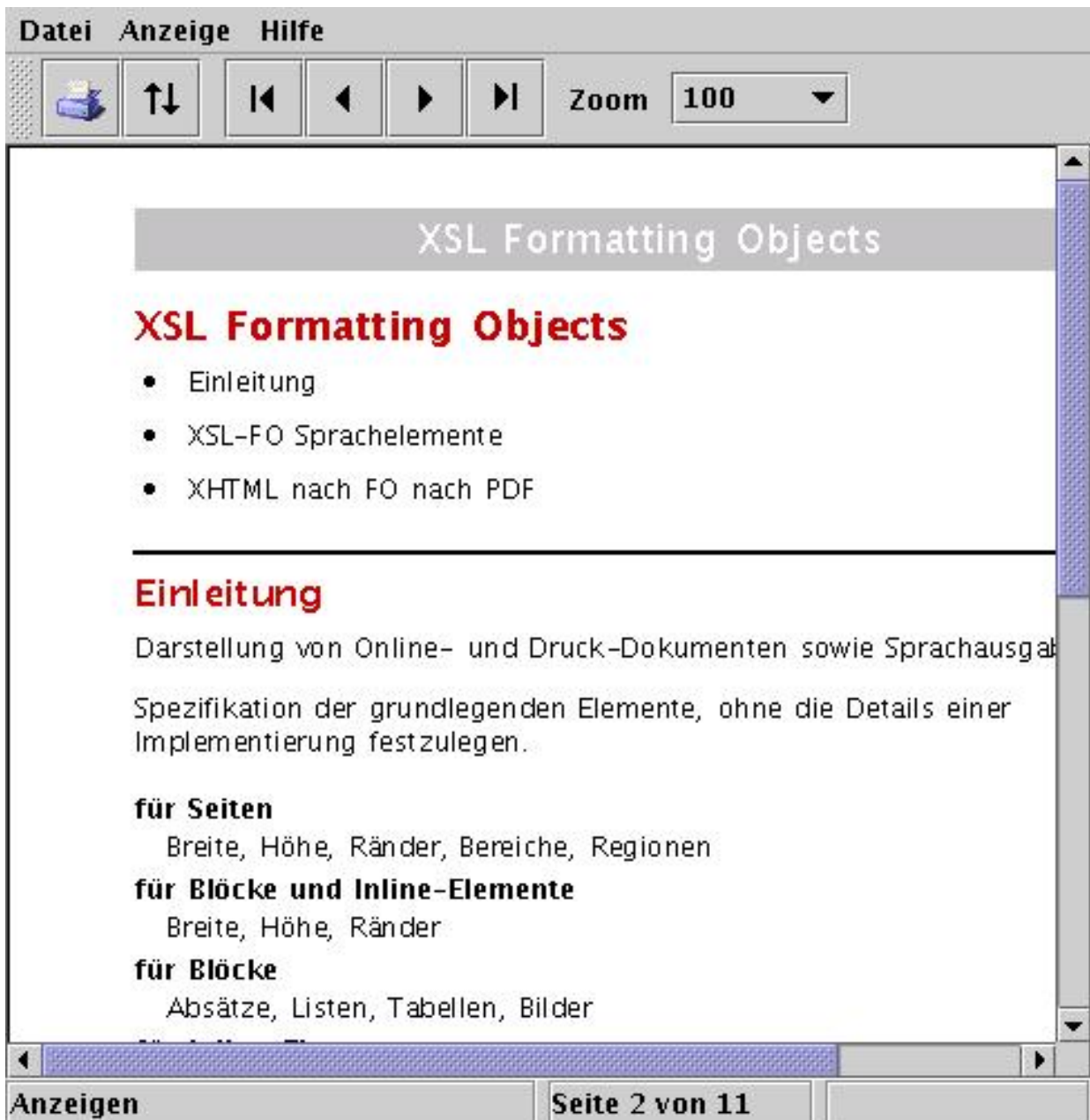
[Ergebnis](#) als PDF.

16.3. XHTML nach FO nach PDF

Es existieren verschiedene Softwarepakete, die XSL-FO verarbeiten können.

Apache Formatting Objects Processor (FOP)

Bietet Konversion von XSL-FO Dateien nach PDF. Kann auch den Vorbereitungsschritt mit XSLT nach XSL-FO mit übernehmen.



Apache FOP AWT-Vorschau

Optionen:

```
./fop.sh
USAGE
Fop [options] [-fo|-xml] infile [-xsl file] [-awt|-pdf|-mif|-pcl|-ps|-txt|-
at|-print] <outfile>
[OPTIONS]
-d          debug mode
-x          dump configuration settings
-q          quiet mode
```

XSL Formatting Objects

```
-c cfg.xml    use additional configuration file cfg.xml
-l lang       the language to use for user information
-s           for area tree XML, down to block areas only

[INPUT]
infile       xsl:fo input file (the same as the next)
-fo infile   xsl:fo input file
-xml infile  xml input file, must be used together with -xsl
-xsl stylesheet xslt stylesheet

[OUTPUT]
outfile      input will be rendered as pdf file into outfile
-pdf outfile input will be rendered as pdf file (outfile req'd)
-awt         input will be displayed on screen
-mif outfile input will be rendered as mif file (outfile req'd)
-pcl outfile input will be rendered as pcl file (outfile req'd)
-ps outfile  input will be rendered as PostScript file (outfile req'd)
-txt outfile input will be rendered as text file (outfile req'd)
-txt.encoding encoding use the encoding for the output file.
              the encoding must be a valid java encoding.
-svg outfile input will be rendered as an svg slides file (outfile req'd)
-at outfile  representation of area tree as XML (outfile req'd)
-print      input file will be rendered and sent to the printer
              see options with "-print help"

[Examples]
Fop foo.fo foo.pdf
Fop -fo foo.fo -pdf foo.pdf (does the same as the previous line)
Fop -xsl foo.xsl -xml foo.xml -pdf foo.pdf
Fop foo.fo -mif foo.mif
Fop foo.fo -print or Fop -print foo.fo
Fop foo.fo -awt
```

Die Vorlesung als PDF

Ein [Beispiel](#) für die Transformation von XHTML nach XSL-FO für den FO-Processor FOP von Apache.org.

Eine Alternative mit mehr Funktionen bietet der Antennahouse XSL Formater AXF.

Die Haupttexte der [Vorlesung](#) als PDF.

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sun May 23 16:50:37 CEST 2004

17. XML Schema

- Einleitung
- XML Schema Sprachkonstrukte: Structure
- XML Schema Sprachkonstrukte: Datatypes
- Beispiele
- Stand der Entwicklung

17.1. Einleitung



- Ersatz und Erweiterung für XML DTD
- (Neben-)bedingungen für Aufbau der Dokumentenstruktur
z.B. Anzahl bestimmter Elemente
- Bedingungen über Datentypen für Elemente und Attribute
z.B. Integer, Float, URL
- Objektorientierung und Vererbung
- mehrere konkurrierende Ansätze:
XDR (XML-Data Reduced, BizTalk), SOX (Schema for Object-Oriented XML), Schematron, DSD (Document Structure Description), XML Schema
- seit Mai 2001 W3C Recommendation
- Beispiel in XML

```
<!ELEMENT KontoStand (#PCDATA) >
```

und XML-Schema

```
<element name="KontoStand">  
<simpleType type="float" />  
</element>
```

Notwendigkeit von Datentypen

```
class Bestellung {  
    Integer lfdNr;  
    String bemerkung;  
    Date bestellDatum;  
    Date lieferDatum;  
    Adresse rechnungsAdresse;  
    Adresse lieferAdresse;  
    Artikel[] artikel;  
}  
  
class Adresse {  
    String vorName;  
    String nachName;  
    String strasse;  
    Integer plz;
```

```
String ort;
}

class Artikel {
    Integer aNr;
    String bezeichnung;
    Float preis;
    Integer anzahl;
}

class Date { ... }
```

Wie können diese Informationen typischer über das Internet ausgetauscht werden?

Mit XML ist nur die folgende Definition möglich.

```
<!ELEMENT Bestellung (lfdNr, bemerkung?,
                      bestellDatum, lieferDatum,
                      rechnungsAdresse, lieferAdresse,
                      waren) >
<!ELEMENT lfdNr (#PCDATA) >
<!ELEMENT bemerkung (#PCDATA) >
<!ELEMENT bestellDatum (#PCDATA) >
<!ELEMENT lieferDatum (#PCDATA) >

<!ELEMENT rechnungsAdresse (vorName, nachName,
                             strasse, plz, ort) >
<!ELEMENT lieferAdresse (vorName, nachName,
                          strasse, plz, ort) >
<!ELEMENT vorName (#PCDATA) >
<!ELEMENT nachName (#PCDATA) >
<!ELEMENT strasse (#PCDATA) >
<!ELEMENT plz (#PCDATA) >
<!ELEMENT ort (#PCDATA) >

<!ELEMENT waren (artikel+) >
<!ELEMENT artikel (aNr, bezeichnung?,
                  preis?, anzahl) >
<!ELEMENT aNr (#PCDATA) >
<!ELEMENT bezeichnung (#PCDATA) >
<!ELEMENT preis (#PCDATA) >
<!ELEMENT anzahl (#PCDATA) >
```

Probleme mit dieser Definition:

- lfdNr, plz, aNr, anzahl sind nur als Zeichenkette definiert, nicht als Integer
- preis ist nur als Zeichenkette definiert, nicht als Float
- bestellDatum, lieferDatum sind nur als Zeichenkette definiert, nicht als echter normalisierter Datumstyp

folgendes fehlerhaftes Dokument lässt sich damit definieren:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Bestellung SYSTEM "Bestellung.dtd" >

<lfdNr>4711</lfdNr>
<bemerkung>eilig</bemerkung>
<bestellDatum>12.1.01</bestellDatum>
<lieferDatum>2001-01-30</lieferDatum>

<rechnungsAdresse>
<vorName>Karl</vorName>
<nachName>Dall</nachName>
<strasse>L 15, 16</strasse>
<plz>68131</plz>
<ort>Mannheim</ort>
</rechnungsAdresse>
```

```
<lieferAdresse>
<vorName>Karl</vorName>
<nachName>Knallinger</nachName>
<strasse>A 5, 6</strasse>
<plz>D-68131</plz>
<ort>Mannheim</ort>
</lieferAdresse>

<waren>
<artikel>
<aNr>4712</aNr>
<anzahl>zwei</anzahl>
</artikel>
<artikel>
<aNr>4713</aNr>
<anzahl>-3</anzahl>
</artikel>
</waren>
</Bestellung>
```

Eine XML Schema Definition mit streng typisierten Definitionen folgen im Abschnitt [Beispiele](#)

17.2. XML Schema Sprachkonstrukte: Structures

von XML zu Schema

in XML DTD:

```
<!ELEMENT ename (cname*,econtent) >
<!ATTLIST ename aname atyp awert >
```

XML Schema Definition:

```
<element name="ename" >
<complexType>
<sequence>
<element name="cname" type="mytype" maxOccurs="5" />
econtent
</sequence>
<attribute name="aname" >
<simpleType type="atyp" value="awert" />
</attribute>
</complexType>
</element>
```

- genauere Kontrolle über die Datentypen im Content
Bedingungen über die Anzahl und der Datentypen: Basis, Derived, eigene
- Kontrolle über die Datentypen und Werte von Attributen
in der gleichen Weise wie für Elemente

Namensräume

- XML Schema Namespaces (Prefix oft xsd:)

```
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
```

- und für XML-Schema Dokumente (Prefix oft xsi:)

```
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
```

- URI für XML-Schema Definitionen

```
xsi:schemaLocation='url1 url2 url3 ...'
```

```
xsi:noNamespaceSchemaLocation='datei.name'
```

XML Schema Processor Conformance

- minimal
für alle Programme und Tools, die Schema Validieren
- representational
Validieren auch externe XML-Repräsentationen von Schema Definitionen
- full
Validieren auch Schema im WWW

XML Schema Datentypen (schema components)

- einfache / elementare Typen (simple types)
- zusammengesetzte Typen (complex types)
- Attribute (attributs)
- Elemente (elements)
- Attribut-Gruppen
- Identity Constraints
key, keyref, unique
- Modell Gruppen
Folgen (sequence), Conjunction (All-Listen), Disjunction (Choice-Listen)
- Notationen
public, system URI
- Annotationen, Bemerkungen, Kommentare
- Partikel (particle, Inhaltsmodell, Inhalt, Wildcards)
- Wildcards (für Namensräume)

XML Schema Facetten (schema facets)

Beschränkungen (constraints) der Datentypen

- length, minLength, maxLength
bei Listen und Zeichenketten
- pattern, enumeration
fast überall
- maxInclusive, maxExclusive, minInclusive, minExclusive
bei Zahlen und Zeiten
- precision, scale
bei Zahlen
- duration, period
bei Zeiten

Schema

```
<schema
  xmlns="http://www.w3.org/2000/10/XMLSchema
  targetNamespace="http://purl.org/metadata/dublin_core"
  version="M.n"
>
Content: ((include | import | redefine | annotation)* ,
  ((attribute | attributeGroup |
    complexType | element | group | notation | simpleType),
    annotation*))
</schema>
```

- Root Element, enthält alle anderen Konstrukte

Element

```
<element
  default = string
  fixed = string
  maxOccurs = for maxOccurs : 1
  minOccurs = nonNegativeInteger : 1
  name = NCName
  type = QName
  ...
>
Content: (annotation? ,
  ((simpleType | complexType)? ,
    (key | keyref | unique)*))
</element>
```

- Element, wie in XML
- Datentypen können per type Attribut referenziert werden
- Datentypen können auch direkt im Content definiert werden

Attribute

```
<attribute
  name = NCName
  type = QName
  use = prohibited | optional | required | default | fixed : optional
  value = string
  ...
>
Content: (annotation? , (simpleType?))
</attribute>
```

- Im Content von Attributen können einfache Datentypen definiert werden
- Datentypen können auch per type Attribut referenziert werden

complexType

```
<complexType
  mixed = boolean : false
  name = NCName
  ...
>
Content: (annotation? ,
  (simpleContent | complexContent |
    ((group | all | choice | sequence)? ,
      ((attribute | attributeGroup)* ,
        anyAttribute?))
  ))
</complexType>
```

- Definition von Datentypen, die komplexeren Content haben können (Folgen, Mengen, Selektionen, Content, Attribute)
- Inhaltsmodell: group | all | choice | sequence
- Inhalt: simpleContent | complexContent
- Attribute: attribute | attributeGroup, anyAttribute

simpleType

```
<simpleType
  name = QName
  ...
>
Content: (annotation?,
  ((list | restriction | union)))
</simpleType>
```

- Definition von einfachen Datentypen, die keine Attribute haben können mit einfachem Content

complexContent

```
<complexContent
  id = ID
  mixed = boolean
  ...
>
Content: (annotation?,
  (restriction | extension))
</complexContent>
```

- Definition des Inhalts eines Elements als Erweiterung oder Einschränkung eines Datentyps
- mixed = true: wenn Text und Elemente gemischt sein dürfen

simpleContent

```
<simpleContent
  id = ID
  ...
>
Content: (annotation?,
  (restriction | extension))
</simpleContent>
```

- Definition eines einfachen Inhalts eines Elements nur als Erweiterung oder Einschränkung eines Datentyps

restriction

```
<restriction
  base = QName
  id = ID
  ...
>
simpleContent: (annotation?,
  (simpleType?,
  (minExclusive | minInclusive | maxExclusive | maxInclusive
  | totalDigits | fractionDigits | length | minLength
  | maxLength | enumeration | whiteSpace | pattern)*)?,
  ((attribute | attributeGroup)*, anyAttribute?))
```

```
complexContent: (annotation?,  
  (group | all | choice | sequence)?,  
  ((attribute | attributeGroup)*, anyAttribute?))  
</restriction>
```

- Einschränkung eines einfachen bzw. komplexen Datentyps
- bei simpleType Facetten
- bei complexType Inhaltsmodell

extension

```
<extension  
  base = QName  
  id = ID  
  ...  
>  
simpleContent: (annotation?,  
  ((attribute | attributeGroup)*, anyAttribute?))  
complexContent: (annotation?,  
  ((group | all | choice | sequence)?,  
  ((attribute | attributeGroup)*, anyAttribute?)))  
</extension>
```

- Erweiterung eines einfachen bzw. komplexen Datentyps
- bei simpleType nur Attribute
- bei complexType Inhaltsmodell

attributeGroup

```
<attributeGroup  
  name = NCName  
  ...  
>  
Content: (annotation? ,  
  ((attribute | attributeGroup)* ,  
  anyAttribute?))  
</attributeGroup>
```

- zur Zusammenfassung von Attributen zu Gruppen

group

```
<group  
  maxOccurs = for maxOccurs : 1  
  minOccurs = nonNegativeInteger : 1  
  name = NCName  
  ...  
>  
Content: (annotation? ,  
  (all | choice | sequence?))  
</group>
```

- zur Zusammenfassung von Elementen zu Gruppen

notation

```
<notation
```

```

    name = NCName
    public = A public identifier, per ISO 8879
    system = uriReference
    ...
  >
Content: (annotation?)
</notation>

```

- Definition von Bezeichnungen

annotation

```

<annotation>
Content: (appinfo | documentation)*
</annotation>

```

- zur Dokumentation der Schema-Definitionen

17.3. XML Schema Sprachkonstrukte: Datatypes

Datentypen Definition

```

name = NCName
variety = ( atomic | list | union )
[baseTypeDefinition]
[facets]
[fundamentalFacets]
[annotation]

```

- Definition der primitiven, abgeleiteten und eigenen Datentypen
- Einschränkung oder Erweiterung des Datentyps durch Facetten
- Facetten sind:
Muster (pattern), Aufzählungen (enumeration), Längenbeschränkungen, Wertebereichsbeschränkungen, Genauigkeit bei Zahlen

Basis Datentypen und Facetten

	{base type definition}	applicable {facets}
If {variety} is list, then		
	[all datatypes]	length, minLength, maxLength, enumeration
If {variety} is union, then		
	[all datatypes]	pattern, enumeration
else if {variety} is atomic, then		
primitive	string	length, minLength, maxLength, pattern, enumeration
boolean	pattern	
float	pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
double	pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	

XML Schema

decimal	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
timeDuration	pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
recurringDuration	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
binary	encoding, length, minLength, maxLength, pattern, enumeration	
uriReference	length, minLength, maxLength, pattern, enumeration	
ID	length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
IDREF	length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
ENTITY	length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
NOTATION	length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
QName	length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
derived	language	length, minLength, maxLength, pattern, enumeration
IDREFS	length, minLength, maxLength, enumeration	
ENTITIES	length, minLength, maxLength, enumeration	
NMTOKEN	length, minLength, maxLength, pattern, enumeration	
NMTOKENS	length, minLength, maxLength, enumeration	
Name	length, minLength, maxLength, pattern, enumeration	
NCName	length, minLength, maxLength, pattern, enumeration	
integer	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
nonPositiveInteger	precision, scale, pattern, enumeration, maxInclusive, maxExclusive,	

	minInclusive, minExclusive	
negativeInteger	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
long	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
int	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
short	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
byte	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
nonNegativeInteger	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
unsignedLong	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
unsignedInt	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
unsignedShort	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
unsignedByte	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
positiveInteger	precision, scale, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
timeInstant	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
time	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
timePeriod	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
date	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
month	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
year	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
century	duration, period, pattern, enumeration,	

	maxInclusive, maxExclusive, minInclusive, minExclusive	
recurringDate	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	
recurringDay	duration, period, pattern, enumeration, maxInclusive, maxExclusive, minInclusive, minExclusive	

Quelle: W3C, 2000

17.4. Beispiele

Beispiel Bestellung (einfach)

Einfache Schema Definition mit den Datentypen `string`, `integer`, `decimal` in Anlehnung an die XML DTD.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

  <xsd:element name="Bestellung">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="lfdNr" type="xsd:string" />
        <xsd:element name="bemerkung" type="xsd:string" />
        <xsd:element name="bestellDatum" type="Date" />
        <xsd:element name="lieferDatum" type="Date" />
        <xsd:element name="rechnungsAdresse" type="Adresse" />
        <xsd:element name="lieferAdresse" type="Adresse" />
        <xsd:element name="waren" type="Waren" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="Adresse">
    <xsd:sequence>
      <xsd:element name="vorName" type="xsd:string" />
      <xsd:element name="nachName" type="xsd:string" />
      <xsd:element name="strasse" type="xsd:string" />
      <xsd:element name="plz" type="xsd:integer" />
      <xsd:element name="ort" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Waren">
    <xsd:sequence>
      <xsd:element name="artikel" type="Artikel" maxOccurs="10" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Artikel">
    <xsd:sequence>
      <xsd:element name="aNr" type="xsd:integer" />
      <xsd:element name="bezeichnung" type="xsd:string" minOccurs="0" />
      <xsd:element name="preis" type="xsd:decimal" minOccurs="0" />
      <xsd:element name="anzahl" type="xsd:integer" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="Date" >
    <xsd:restriction base="xsd:string" />
  </xsd:simpleType>
</xsd:schema>
```

XML Schema

```
</xsd:simpleType>
</xsd:schema>
```

Beispiel-Dokument mit Fehlern.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Bestellung xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation='Bestellung-1.xsd'>

  <lfdNr>4711</lfdNr>
  <bemerkung>eilig</bemerkung>
  <bestellDatum>12.1.01</bestellDatum>
  <lieferDatum>2001-01-30</lieferDatum>

  <rechnungsAdresse>
    <vorName>Karl</vorName>
    <nachName>Dall</nachName>
    <strasse>L 15, 16</strasse>
    <plz>68131</plz>
    <ort>Mannheim</ort>
  </rechnungsAdresse>

  <lieferAdresse>
    <vorName>Karl</vorName>
    <nachName>Knallinger</nachName>
    <strasse>A 5, 6</strasse>
    <plz>D-68131</plz>
    <ort>Mannheim</ort>
  </lieferAdresse>

  <waren>
    <artikel>
      <aNr>4712</aNr>
      <anzahl>zwei</anzahl>
    </artikel>
    <artikel>
      <aNr>4713</aNr>
      <anzahl>-3</anzahl>
    </artikel>
  </waren>

</Bestellung>
```

Ohne Fehler:

```
valid Bestellung-1e.xml
Bestellung-1e.xml: 1225 ms (24 elems, 0 attrs, 34 spaces, 104 chars)
```

schema validation script:

```
#!/bin/sh
VALIDPATH="/home/kredel/java/lib/xmlParserAPIs.jar:/home/kredel/java/lib/xercesImpl.jar:/home/kredel/java/"
export CLASSPATH="$VALIDPATH:$CLASSPATH"
/usr/lib/jdk1.3/bin/java sax.Counter -v -s $*
```

Gefundene Fehler:

```
schema Bestellung-1e.xsi
[Error] Bestellung-1e.xsi:15:19: cvc-type.3.1.3:
  The value 'D-68131' of element 'plz' is not valid.
[Error] Bestellung-1e.xsi:30:22: cvc-type.3.1.3:
  The value 'zwei' of element 'anzahl' is not valid.
Bestellung-1e.xsi: 1383 ms (24 elems, 1 attrs, 0 spaces, 138 chars)
```

Die Datumsprobleme und die Anzahl -3 werden nicht entdeckt.

Beispiel Bestellung (streng)

Alle Beispiele für Apache xerces in aktueller Syntax

Verwendung von date

```
<?xml version="1.0" encoding="UTF-8" ?>
<Bestellung xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation='Bestellung-2.xsd'>

  <lfdNr>4711</lfdNr>
  <bemerkung>eilig</bemerkung>
  <bestellDatum>12.1.01</bestellDatum>
  <lieferDatum>2001-01-30</lieferDatum>

  <rechnungsAdresse>
    <vorName>Karl</vorName>
    <nachName>Dall</nachName>
    <strasse>L 15, 16</strasse>
    <plz>68131</plz>
    <ort>Mannheim</ort>
  </rechnungsAdresse>

  <lieferAdresse>
    <vorName>Karl</vorName>
    <nachName>Knallinger</nachName>
    <strasse>A 5, 6</strasse>
    <plz>68131</plz>
    <ort>Mannheim</ort>
  </lieferAdresse>

  <waren>
    <artikel>
      <aNr>4712</aNr>
      <preis>DM 22</preis>
      <anzahl>2</anzahl>
    </artikel>
    <artikel>
      <aNr>4713</aNr>
      <anzahl>3</anzahl>
    </artikel>
  </waren>

</Bestellung>
```

Schema Definition:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

  <xsd:element name="Bestellung">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="lfdNr" type="xsd:string" />
        <xsd:element name="bemerkung" type="xsd:string" />
        <xsd:element name="bestellDatum" type="Date" />
        <xsd:element name="lieferDatum" type="Date" />
        <xsd:element name="rechnungsAdresse" type="Adresse" />
        <xsd:element name="lieferAdresse" type="Adresse" />
        <xsd:element name="waren" type="Waren" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="Adresse">
    <xsd:sequence>
      <xsd:element name="vorName" type="xsd:string" />
```

```
<xsd:element name="nachName" type="xsd:string" />
<xsd:element name="strasse" type="xsd:string" />
<xsd:element name="plz" type="xsd:integer" />
<xsd:element name="ort" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Waren">
<xsd:sequence>
<xsd:element name="artikel" type="Artikel" maxOccurs="10" />
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Artikel">
<xsd:sequence>
<xsd:element name="aNr" type="xsd:integer" />
<xsd:element name="bezeichnung" type="xsd:string" minOccurs="0" />
<xsd:element name="preis" type="xsd:decimal" minOccurs="0" />
<xsd:element name="anzahl" type="xsd:integer" />
</xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="Date" >
<xsd:restriction base="xsd:date" />
</xsd:simpleType>

</xsd:schema>
```

Gefundene Fehler (Apache xerces):

```
schema Bestellung-2e.xsi
[Error] Bestellung-2e.xsi:7:37: cvc-type.3.1.3:
    The value '12.1.01' of element 'bestellDatum' is not valid.
[Error] Bestellung-2e.xsi:29:21: cvc-type.3.1.3:
    The value 'DM 22' of element 'preis' is not valid.
Bestellung-2e.xsi: 1402 ms (25 elems, 1 attrs, 0 spaces, 138 chars)
```

Verwendung von pattern und nonNegativeInteger

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

<xsd:element name="Bestellung">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="lfdNr" type="xsd:string" />
<xsd:element name="bemerkung" type="xsd:string" />
<xsd:element name="bestellDatum" type="Date" />
<xsd:element name="lieferDatum" type="Date" />
<xsd:element name="rechnungsAdresse" type="Adresse" />
<xsd:element name="lieferAdresse" type="Adresse" />
<xsd:element name="waren" type="Waren" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:complexType name="Adresse">
<xsd:sequence>
<xsd:element name="vorName" type="xsd:string" />
<xsd:element name="nachName" type="xsd:string" />
<xsd:element name="strasse" type="xsd:string" />
<xsd:element name="plz" type="xsd:integer" />
<xsd:element name="ort" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Waren">
<xsd:sequence>
<xsd:element name="artikel" type="Artikel" maxOccurs="10" />
```

```

</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Artikel">
<xsd:sequence>
<xsd:element name="aNr" type="xsd:integer" />
<xsd:element name="bezeichnung" type="xsd:string" minOccurs="0" />
<xsd:element name="preis" type="xsd:decimal" minOccurs="0" />
<xsd:element name="anzahl" type="xsd:nonNegativeInteger" />
</xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="Date" >
<xsd:restriction base="xsd:string" >
<xsd:pattern value="\d{4}-\d{2}-\d{2}" />
</xsd:restriction>
</xsd:simpleType>

</xsd:schema>

```

Gefundene Fehler (Apache xerces):

```

schema Bestellung-3e.xsi
[Error] Bestellung-3e.xsi:7:37: cvc-type.3.1.3:
The value '12.1.01' of element 'bestellDatum' is not valid.
[Error] Bestellung-3e.xsi:33:20: cvc-type.3.1.3:
The value '-3' of element 'anzahl' is not valid.
Bestellung-3e.xsi: 1479 ms (24 elems, 1 attrs, 0 spaces, 133 chars)

```

Definition von Elementen mit Text-Inhalt und Attribut.

```

<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

<xsd:element name="Bestellung">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="lfdNr" type="xsd:string" />
<xsd:element ref="bemerkung" />
<xsd:element name="bestellDatum" type="Date" />
<xsd:element name="lieferDatum" type="Date" />
<xsd:element name="rechnungsAdresse" type="Adresse" />
<xsd:element name="lieferAdresse" type="Adresse" />
<xsd:element name="waren" type="Waren" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
...
<xsd:element name="bemerkung" >
<xsd:complexType>
<xsd:simpleContent>
<xsd:extension base="xsd:string" >
<xsd:attribute name="priority" type="xsd:string" />
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>

</xsd:schema>

```

Beispiel Personaldaten

[xml](#), [dtd](#), [xsd](#) (veraltete Syntax)

17.5. Stand der Entwicklung

14. Januar 2002

- XML Schema, Part 0 Primer, Part 1 Structures, Part 2 Datatypes, W3C Recommendation, 2. Mai 2001
- Tools:
Apache Xerces mit SAX und DOM Parser
Java API for XML Processing (JAXP)
Oracle xmlschema

© Universität Mannheim, Rechenzentrum, 1998-2004.

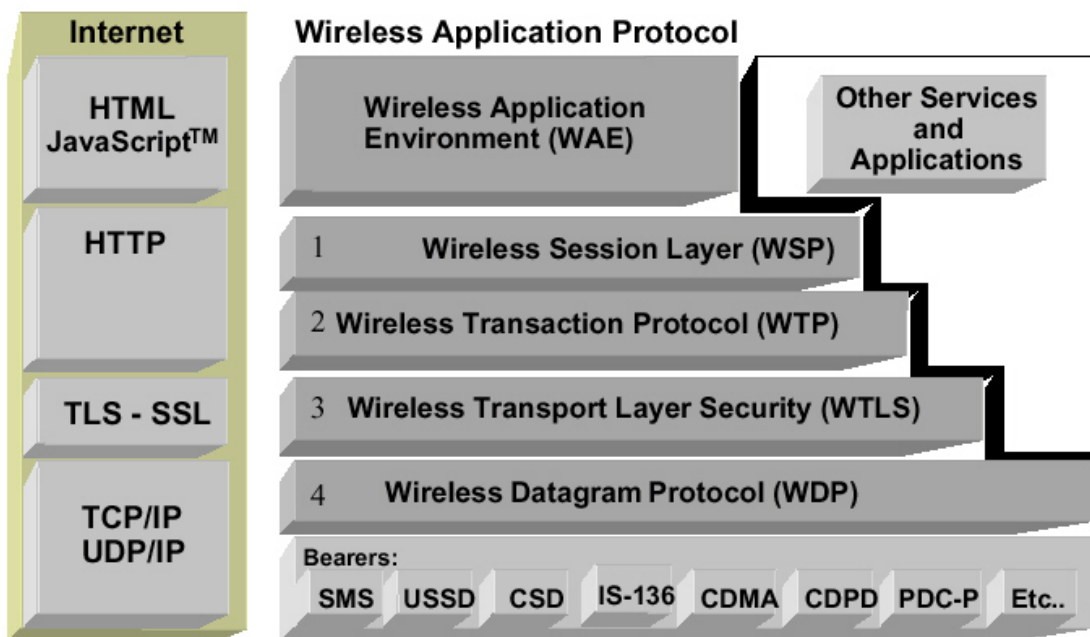
Last modified: Sat May 29 12:21:37 CEST 2004

18. Wireless Markup Language (WML)

- WAP Überblick
- WML Überblick
- WML Sprache

18.1. WAP Überblick

- WAP = Wireless Application Protocol.
- Offene und globale Spezifikation um Benutzern Informationen und Dienstleistungen auf mobilen Endgeräten anbieten zu können.
- WAP spezifiziert wie die Übertragung des Inhaltes auf die Endgeräte erfolgt.
- Diese Endgeräte können Mobiltelefone, Organizer, Palmtops, Pager, Autoradios und Ähnliches sein.



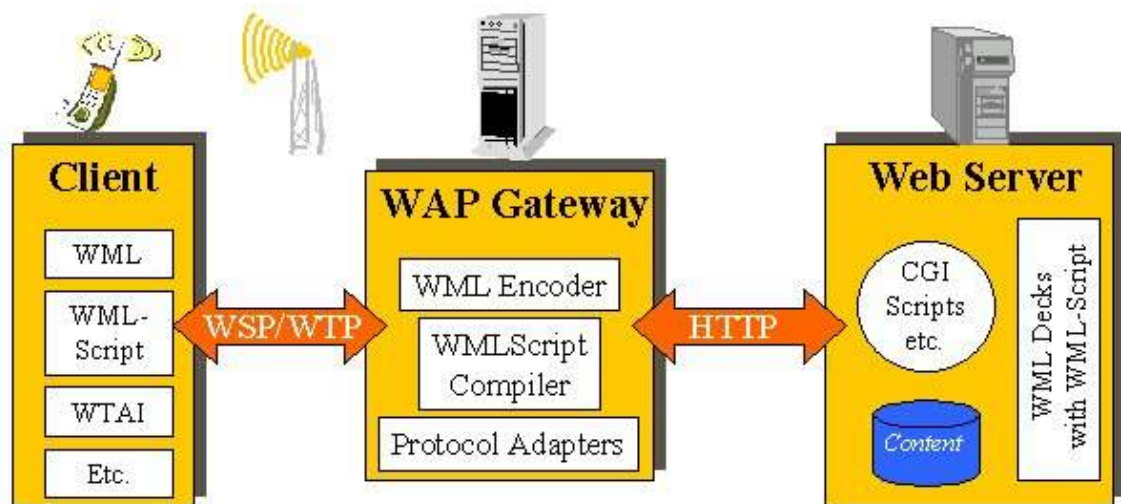
WAP and Internet protocol stacks.

- WAP wird vom WAP Forum entwickelt
Organisation von 400+ Telekommunikationsunternehmen
- Ziel: offener Standard für mobile Internet-Dienste zu schaffen, um eine weltweite Interoperabilität von drahtlosen Diensten sicherzustellen.
- WAP ist nicht an einen bestimmten Mobilfunkstandard wie GSM oder UMTS gebunden, d.h. verschiedene Bearers werden unterstützt
- großen Hersteller zur Unterstützung des WAP-Standards verpflichtet

- WAP kooperiert mit dem W3C bei der Weiterentwicklung der Markupsprachen
- Alternativen zu WAP:
Mobile IP und i-Mode

Ablauf von WAP Verbindungen

The WAP Architecture



©1999 Wireless Application Forum, Ltd.



Eine Anfrage wird im WAP-Modell ist ähnlich einer im Web-Modell. Ein Anfrage könnte wie folgt aussehen:

1. Der Anwender gibt in seinem Handy die URL ein, oder wählt einen vorbereiteten Link.
2. Der WAP Client sendet über das WAP-Protokol eine Anfrage nach dieser URL an ein WAP Gateway.
3. Das WAP Gateway erzeugt eine normale HTTP Anfrage nach der gewünschten URL und sendet diese an den Web Server.
4. Die HTTP Anfrage wird von dem Web Server verarbeitet.
Die URL kann sich auf eine statische Seite beziehen oder auf eine CGI oder sonstige Script Anwendung.
5. Der Web Server gibt den WML Deck mit dem hinzugefügten HTTP Header die von CGI oder der Script-Anwendung erzeugten WML Inhalt zurück.
6. Das WAP Gateway überprüft den HTTP Header und den WML Inhalt und codiert diesen in einem WAP eigenen Binärformat.
Anschließend kreiert das Gateway die WAP-Antwort mit dem WMLX-Inhalt und schickt diese an den WAP-Client.
7. Der WAP-Client empfängt die WAP-Antwort und verarbeitet den WML-Inhalt.
Der Inhalt der ersten Card des WML Decks wird im Handydisplay angezeigt.

Um auf einem Web-Server WAP-Inhalte präsentieren zu können, müssen nur zwei Einstellungen vorgenommen werden:

text/vnd.wml Dateiendung .wml

image/vnd.wbmp Dateiendung .wbmp

Ausstattung der UAs

User Agents sind vor allem Handys (Cell Phone) und PDAs (Personal Digital Assistant).

Da die Anzeigemöglichkeiten etwa auf Handys und PDAs sehr beschränkt sind gibt es hier einiges Grundlegendes zu beachten.

- Eingeschränktes Display von nur 4-6 Zeilen mit je 10-20 Zeichen. Bei PDAs 30-50 Zeilen mit 40-60 Zeichen.
- Einschränkungen bei der Verwendung von Graphiken.
- So können grundsätzlich nur Bilder mit einer Farbtiefe von 1 Bit (schwarz-weiß) verwendet werden.
- Die maximale Abmessung der darstellbaren Graphiken ist von den verwendeten Endgeräten abhängig
Nokia 7110: 84 x 84 Pixel, Palm III: 160 x 160 Pixel.
- Eingeschränkte Eingabemöglichkeiten, zum Beispiel gibt es keine Maus.
- Langsame CPU mit wenig Hauptspeicher.
- Man sollte außerdem immer berücksichtigen, daß die Übertragungsraten zum Handy derzeit nur bei ca. 9600 Bit/sec (1200 Bytes/sec) liegen, was ebenfalls eine Verwendung von aufwendigen Graphiken wenig sinnvoll macht.
- Latenzzeiten von 5-10 Sekunden, das ist die Zeit zwischen dem Beginn einer Anfrage und dem Eintreffen der Antwort.

Auf keinen Fall sollte man versuchen bestehende HTML-Seiten einfach in WML umzuwandeln. Sondern man sollte versuchen eine sinnvolle Informations-Reduzierung vorzunehmen. Erfahrungsberichte zeigen, daß dies der mit Abstand schwierigste und zeitaufwendigste Teil der Erstellung von WAP-Angeboten ist.

18.2. WML Übersicht

WML steht für Wireless Markup Language.

Es ist eine offene Sprache, mit einer XML DTD, in der einige Elemente von HTML mit neuen Elementen kombiniert werden.

Stand der Entwicklung

- WML Version 1.0, 30. April 1998
- WML Version 1.1, 16. Juni 1999
- WML Version 1.2, 4. November 1999
- WML Version 1.3, 19. Februar 2000

Kleines WML Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="card1" title="Title">
<p>
    Ein einfacher WML-Deck!!!
</p>
</card>
</wml>
```

Zeilenweise Erklärung:

1. Die erste Zeile gibt die verwendete XML Version an.
2. In der zweiten Zeile steht welche DTD verwendet wird.

3. In der dritten Zeile steht der für alle WML Dokumente obligatorische <wml> Tag.
4. In der vierten Zeile steht der Card Identifier, mit der eine Card eines Decks explizit angesprochen werden kann und ein Titel.
5. In der fünften Zeile wird ein ebenfalls obligatorisches Paragraph Element geöffnet.
6. In der sechsten Zeile steht der auszugebende Text
7. In der siebten Zeile wird das Paragraph Element geschlossen.
8. In der achten Zeile wird das Card Element geschlossen.
9. In der neunten Zeile wird das WML Element geschlossen.

Das Ergebnis:



Ansicht mit dem Nokia WAP-Toolkit 1.2

Das dargestellte Handy ist das Nokia 6150

WML und XML

Eine WML Datei muss sich als XML Anwendung mit DTD zu erkennen geben.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

Da WML eine XML Anwendung ist gilt insbesondere:

- die Elemente und Attribute sind case-sensitiv
- es werden meist keine Umlaute unterstützt, obwohl WAP 1.1 das vorsieht
- Start- und End-Tag wie für XML Anwendungen
- Schachtelung der Elemente wie für XML Anwendungen
- Aufbau und Verwendung der URLs wie bei HTML
Fragmentidentifizier (#), relative URLs

Die sonstigen syntaktischen Konstrukte in WML sind die, die in XML definiert werden.

Zur Erinnerung werden die Sprachelemente Entities, Tags, Elemente und Attribute im Folgenden kurz wiederholt.

Entities: Zur Strukturierung von Dokumenten werden in XML-basierten Sprachen Speicherungseinheiten verwendet, die so genannten Entities. Ein XML-Dokument kann aus einer Entity oder mehreren Entities bestehen. Über diese Speicherungseinheiten wird ein Dokument in mehrere voneinander unabhängig Teilkomponenten aufgespalten, die einzeln bearbeitet werden können. In WML werden Entities insbesondere für die Spezifikation der Zeichen verwendet. So wird beispielsweise das Und-Zeichen (&) durch das Entity & repräsentiert. Entities beginnen immer mit dem Und-Zeichen und enden mit einem Strichpunkt.

Tags: Auch in HTML werden Tags in der gleichen Form benutzt. Ein Tag beschreibt ein Element und beinhaltet den Elementnamen und einen eindeutigen Identifier. Ein Tag kann zudem Attribute enthalten, die weitere Eigenschaften eines Elements spezifizieren. Auch bei WML gibt es Start-, End- und leere Tags.

Tags werden grundsätzlich klein geschrieben.

Elemente: Elemente spezifizieren Markierungen und Strukturinformationen eines WML-Decks. Sie weisen eine von zwei Strukturen auf:

Bei nicht-leeren Elementen wird die eigentliche Information von Anfangs- und End-Tag eingefasst. Leere Elemente wie beispielsweise
 für den Zeilenumbruch besitzen keinen Content. Elemente werden je nach Funktionalität in Kategorien eingeteilt. So unterscheidet man beispielsweise zwischen Textformatierungen und Event-basierten Elementen.

Attribute: Viele WML-Elemente können mit Attributen versehen werden, die zusätzliche Informationen für ein Element bestimmen. Attribute werden immer im Start-Tag eines Elements spezifiziert.

Attribut-Wert-Paare werden durch ein Leerzeichen voneinander getrennt. Die Werte müssen in Anführungszeichen gesetzt werden. Für den Attributnamen ist Kleinschreibung zwingend vorgeschrieben. Bei einigen Elementen ist das zugehörige Attribut festgelegt. Beispielsweise verlangt das Element go das Attribut href.

Andere Attribute wiederum sind optional oder aber besitzen Standardwerte. Für das img-Element kann beispielsweise das Attribut align verwendet werden. Ist kein Attribut angegeben, so wird der Standardwert bottom verwendet.

Weiteres WML Beispiel

Ein Beispiel mit 2 cards:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

```
<wml>
<card id="card1" title="Semia">
<p>
  Ein einfacher WML-Deck!!!
<br />
<a href="#card2">2. card</a>
</p>
</card>
<card id="card2" title="Semib">
<p>
  Noch eine WML-Card!!!
<br />
<a href="#card1">1. card</a>
</p>
</card>
</wml>
```

Das Ergebnis:



Ansicht mit dem Nokia WAP-Toolkit 1.2

18.3. WML Sprache

Neue ELe mente in WML

WML unterscheidet sich von HTML in der Art und Weise, wie Inhalte Strukturiert sind. Während HTML seitenorientiert arbeitet, werden WAP-Inhalte in so genannte Cards und Decks zerlegt.

Bei den **Cards** handelt es sich um die grundlegenden Einheiten von WML.

Eine Card ist die Einheit, die auf dem Display dargestellt wird, außerdem ist in ihnen die Interaktion zwischen dem Browser und dem Benutzer definiert.

In Cards werden beispielsweise Auswahl- oder Eingabefelder definiert.

Mehrere Cards werden zu einem **Deck** zusammengefasst.

Decks sind somit die übergeordneten Elemente einer WAP-Anwendung.

Ein Deck wird mit allen darin enthaltenen Cards komplett an den Client übertragen.

Erreicht ein Deck den WAP-Client, so wird in der Regel die erste Card geöffnet - es sei denn, über den Client wird explizit eine andere Card angesprochen.

Neu gegenüber HTM sind **Variablen**, die in Elementen (in PCDATA) und Attributen (in VDATA) verwendet werden können. Syntax:

```
$identifizier
$(identifizier )
$(identifizier : conversion )
```

Das \$ Escape-Zeichen ist wieder \$. D.h. \$\$ steht für ein Text \$. Mit conversion können Umlaute etc. in URLs kodiert werden.

Variablen können mit dem setvar Element Werte zugewiesen werden. Die Auswertung der Werte erfolgt auf dem UA.

```
<setvar name="varname" value="wert" />
```

wert darf natürlich selbst wieder Variablen enthalten.

WML Elemente und Attribute

Die wichtigsten WML Elemente werden im Folgenden besprochen.

Textformatierung

- White space Behandlung wie bei XML.
- Hervorhebungen:
em, strong, big, small
i, b, u wie in HTML
- Paragraphen:
p, br wie in HTML
- Tabellen:
table, tr, td wie in HTML

Bilder und Anchors

- Bilder:
img wie in HTML
- Anchor:
a wie in HTML

User input

- es gibt kein form-Element
zur Gruppierung von Feldern und Text kann `fieldset` benutzt werden
zur Gruppierung von Optionen kann `optgroup` benutzt werden
- Eingabefelder:
`input` ähnlich wie HTML
neues Attribut `format` definiert das akzeptierte Eingabeformat
`A, a, N, X, x, M, m, *f, nf, \c`
neues Attribut `emptyok` falls das Feld leer sein darf
- Auswahlfelder:
`select` ähnlich wie HTML
- Auswahloptionen:
`option` ähnlich wie HTML

Decks und Cards

- WML Root-Element `wml`
Content-Modell: `head? template? card+`
- eine Anzeigeeinheit `card`
Content-Modell: `onevent*, timer?, (do | p)*`
Attribute: `title, newcontext, ontimer, onenterforward, onenterbackward`
- Mustereinheit `template`
Definition von Elementen, die in jeder folgenden Card des Decks gelten
- Kopfteil `head`
Content-Modell: `access meta+`
- Zugriff `access`
`<access domain="dom" path="p" >`
- Metadaten `meta` wie in HTML
- Reaktion auf Benutzereingabe `do`
Content-Modell: `go | prev | noop | refresh`
Attribute: `type, label`
Darstellung durch UA Widget (zB Handy Taste)

Events

- das `type` Attribut steuert die Bedeutung von `do`
- Bejahung: `accept`
- Zurück: `prev`
- Hilfe: `help`
- Zurücksetzen: `reset`
- weitere Optionen: `options`
- Löschen: `delete`
- Leer: `unknown`
- Experimentelle Anwendung: `X-name`
- Herstellerspezifische Anwendung: `VND.name`
- Timer Element: `timer`
Attribut: `value="sec/10"`
- Eingebaute Ereignisse

- Timer Ablauf: `ontimer`
- beim Eingang zur Card: `onenterforward`
- beim Ausgang aus der Card: `onenterbackward`
- beim Selektieren eines Elements: `onpick`
- Binden von Elementen zu Ereignissen `onevent`

Tasks

- Navigation zu URL:
`<go href="url" method="get|post">`
Content-Modell: `(postfield | setvar)*`
ähnlich wie `<form action="url">` in HTML.
- für die Definition des Querystrings:
`<postfield name="x" value="v" />`
- setzen einer Variable:
`<setvar name="x" value="v" />`
- als Anchor verwenden:
`<anchor>text <go href="url" /></anchor>`
entspricht
`<a href="url" >text</a>`
- Zurück: `prev`
- Neu laden: `refresh`
- Leer: `noop`

WML Beispiele

Die folgenden Beispiele lassen sich u.A. mit dem [Opera Browser](#) ansehen.

- [ein Deck](#)
- [zwei Decks](#)
- [Listen, Tabellen, Timer](#)
- [Buttons, Auswahl und Eingabe](#)
- [Templates und Variablen](#)

18.4. Ausblick

- WBXML: WAP Binary XML Content Format
Komprimierte Übertragung der Daten zum Handy
 - WMLScript ähnlich wie JavaScript
 - WML 1.2 und 1.3
 - Server-Push: Events, die vom Server auf dem Client generiert werden können
 - WMLScript Crypto Library
 - CC/PP
Composite Capability / Preference Profile Framework
 - Kooperation WAPForum mit W3C
-

Erstellt unter Verwendung eines Seminarvortrags von Bertram Schmitt und der WML 1.1 Spezifikation.

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:38:57 CEST 2004

19. Java API for XML Processing (JAXP)



Parser = Programm zum Einlesen von Dokumenten entsprechend einer Syntax-Definition. Das Ergebnis ist eine interne Datenstruktur.

- Lexikalische Analyse: Scanner
z.B. LEX
- Syntaktische Analyse:
z.B. YACC oder GNU BISON,
bei XML Wohlgeformtheit feststellen
- Semantische Analyse:
auch YACC / BISON,
bei XML Gültigkeit bez. DTD feststellen

Ausblick: Java API for XML Binding (JAXM)
Generation der Java Klassen direkt aus einer DTD.

19.1. DOM und SAX Parsers

- DOM = Document Object Model, W3C
- erzeugt (Java-)Objekt aus XML Datei, erlaubt Manipulation dieser Objekte
- DocumentBuilder
- DocumentBuilderFactory
- SAX = Simple API for XML, XML-Developer
- Ereignisse (wie startElement) werden durch Handler verarbeitet
- SAXParser
- SAXParserFactory
- SAX 2 mit Namespace Unterstützung

Vorteile:

- DOM liefert fertiges Dokumenten Objekt
- SAX ist sehr effizient und braucht wenig Speicher

Nachteile:

- DOM ist ineffizient und braucht viel Speicher
- bei DOM muss evtl. die Datenstruktur doppelt erzeugt werden

- bei SAX keine Garantie, dass gültige Dokumente entstehen
- Entwicklung der SAX Handler für komplizierte DTDs schwer

19.2. DOM Parser

- Apache Xerces: `org.apache.xerces.parser.DOMParser`
- im JDK ab 1.4 als: `javax.xml.parsers.DocumentBuilder` und `javax.xml.parsers.DocumentBuilderFactory`

DocumentBuilderFactory

`javax.xml.parsers.DocumentBuilderFactory` dient der Erzeugung eines DOM Parsers
`javax.xml.parsers.DocumentBuilder`.

```
protected DocumentBuilderFactory()
public static newInstance()

public DocumentBuilder newDocumentBuilder()

public boolean isValidating()
public void setValidating(boolean)

public boolean isExpandEntityReferences()
public void setExpandEntityReferences(boolean)
```

DocumentBuilder

`javax.xml.parsers.DocumentBuilder` dient dem Parsen, d.h. dem Einlesen und Aufbau eines `org.w3c.dom.Document`.

```
protected DocumentBuilder()
public Document newDocument()

public Document parse(File f)
public Document parse(InputStream is)
public Document parse(InputSource is)
public Document parse(String uri)

public boolean isValidating()
public void setErrorHandler(ErrorHandler)
```

DOMLeser:

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.IOException;
import org.xml.sax.SAXException;

public class DOMLeser {

    public static void main(String[] args) {

        DocumentBuilderFactory pfac = DocumentBuilderFactory.newInstance();
        DocumentBuilder parser = null;

        try {
```

```

        pfac.setValidating(true);
        // pfac.setCoalescing(true);
        pfac.setExpandEntityReferences(false);
        parser = pfac.newDocumentBuilder();
    }
    catch (ParserConfigurationException e) {
        e.printStackTrace();
    }

    Document doc = null;
    String uri = "datei.xml";

    try {
        doc = parser.parse(uri);
    }
    catch (SAXException e) {
        e.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }

    Screen sc = new Screen();
    sc.println("doc(" + doc.getClass().getName() + ") = " + doc );

    DOMSchreiber dw = new DOMSchreiber();
    sc.println("document:\n");
    dw.printNode(doc);
    dw.flush();
    dw.close();
}
}

```

DOM Node

Die Werte von nodeName, nodeValue, und attributes hängen wie folgt vom Knotentyp ab:

Interface	nodeName	nodeValue	attributes
Attr	name of attribute	value of attribute	null
CDATASection	"#cdata-section"	content of the CDATA Section	null
Comment	"#comment"	content of the comment	null
Document	"#document"	null	null
DocumentFragment	"#document-fragment"	null	null
DocumentType	document type name	null	null
Element	tag name	null	NamedNodeMap
Entity	entity name	null	null
EntityReference	name of entity referenced	null	null
Notation	notation name	null	null
ProcessingInstruction	target	entire content excluding the target	null
Text	"#text"	content of the text node	null

Siehe [Document Object Model \(DOM\) Level 2 Core Specification](#).

DOMSchreiber:

```

import org.w3c.dom.Document;
import org.w3c.dom.Node;

```

```

import org.w3c.dom.NodeList;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Element;
import org.w3c.dom.Text;
import org.w3c.dom.Comment;
import org.w3c.dom.EntityReference;
import org.w3c.dom.DocumentType;
import java.io.IOException;
import java.io.PrintWriter;
import org.xml.sax.SAXException;

public class DOMSchreiber {

    PrintWriter pw = null;

    public DOMSchreiber() {
        pw = new Screen();
    }

    public DOMSchreiber(PrintWriter p) {
        pw = p;
    }

    public void printNode(Node node) {
        switch ( node.getNodeType() ) {
            case Node.ELEMENT_NODE:
                printElement( (Element)node );
                break;
            case Node.ATTRIBUTE_NODE:
                printAttributes( (NamedNodeMap)node );
                break;
            case Node.TEXT_NODE:
                printText( (Text)node );
                break;
            case Node.CDATA_SECTION_NODE:
                printCDATA( (Text)node );
                break;
            case Node.PROCESSING_INSTRUCTION_NODE:
                printPI( node );
                break;
            case Node.COMMENT_NODE:
                printComment( (Comment)node );
                break;
            case Node.DOCUMENT_TYPE_NODE:
                printDoctype( (DocumentType)node );
                break;
            case Node.DOCUMENT_NODE:
                printDocument( (Document)node );
                break;
            case Node.ENTITY_REFERENCE_NODE:
                printEntityReference( (EntityReference)node );
                break;
            default: pw.println("<!-- " + node + " -->");
        }
    }

    public void printElement(Element el) {
        String name = el.getNodeName();
        pw.print("<" + name );
        printAttributes( el.getAttributes() );
        NodeList nl = el.getChildNodes();
        if ( ( nl == null ) || ( nl.getLength() == 0 ) ) {
            pw.print(" />");
            return;
        }
        pw.print(">");
        for (int i = 0; i < nl.getLength(); i++) {
            printNode(nl.item(i));
        }
        pw.print("</" + name + ">");
    }
}

```

```

}

public void printAttributes(NamedNodeMap at) {
    if ( at == null ) return;
    for ( int i = 0; i < at.getLength(); i++ ){
        Node an = at.item(i);
        pw.print(" " + an.getNodeName() + "=\""");
        pw.print(an.getNodeValue() + "\"");
    }
}

public void printText(Text tx) {
    pw.print( tx.getData() );
}

public void printEntityReference(EntityReference er) {
    pw.print("&" + er.getNodeName() + ";" );
}

public void printComment(Comment c) {
    pw.print("<!--");
    pw.print( c.getData() );
    pw.print("-->");
}

public void printDoctype(DocumentType d) {
    pw.print("<!DOCTYPE ");
    pw.print( d.getName() );
    String pid = d.getPublicId();
    if ( pid != null ) {
        pw.print(" PUBLIC \"" + pid + "\"\n          ");
    } else {
        pw.print(" SYSTEM " );
    }
    pw.print("\"" + d.getSystemId() + "\" ");
    pw.println(">");
}

public void printCDATA(Text tx) {
    pw.print("<![CDATA[ ");
    pw.print( tx.getNodeValue() );
    pw.print("]]>");
}

public void printDocument(Document doc) {
    printNode( doc.getDoctype() );
    printNode( doc.getDocumentElement() );
}

public void printPI(Node pi) {
    pw.print("<?" + pi.getNodeName() );
    pw.print(" " + pi.getNodeValue() + "?>" );
}

public void flush() {
    pw.flush();
}

public void close() {
    pw.close();
}

```

```
}
```

datei.xhtml:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "DTD/xhtml11-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Titel</title>
<?php
    $a=9;
    function yyy(a) {
        if ( a > 2 ) {
            print("<p>nonsense</p>");
        }
    }
?>
<!-- funktions definitionen -->
<script type="text/javascript" language="JavaScript">
<![CDATA[
    function xxx(a) {
        if ( a > 2 ) {
            document.write("<p>nonsense</p>");
        }
    }
]]>
</script>
</head>
<body bgcolor="white">
<h1>Überschrift A</h1>
<p align="center">Paragraph mit <em>Hervorhebung</em>
und einem <br /> Zeilenumbruch.
</p>
<ul>
<li>Listen Element</li>
<li>Entity Element &abc; </li>
</ul>
<h1>Überschrift B</h1>
<p>Text eines zweiten Paragraphen mit
<strong>Hervorhebung</strong>
ohne Zeilenumbruch.
</p>
<h1 count="10">Überschrift C</h1>
<p>Text eines dritten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
<h1>Überschrift D</h1>
<p>Text eines vierten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
</body>
</html>
```

Ausgabe:

```
Warning: validation was turned on but an org.xml.sax.ErrorHandler was not
set, which is probably not what is desired. Parser will use a default
ErrorHandler to print the first 10 errors. Please call
the 'setErrorHandler' method to fix this.
Error: URI=datei.xhtml Line=41: Attribute "count" must be declared for element type "h1".
doc(org.apache.xerces.dom.DeferredDocumentImpl) = [#document: null]
document:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "DTD/xhtml11-transitional.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Titel</title>
```

```

<?php $a=9;
function yyy(a) {
    if ( a > 2 ) {
        print("<p>nonsense</p>");
    }
}
?>
<!-- funktions definitionen -->
<script language="JavaScript" type="text/javascript">
<![CDATA[ ]]>
</script>
</head>
<body bgcolor="white">
<h1>Überschrift A</h1>
<p align="center">Paragraph mit <em>Hervorhebung</em>
und einem <br clear="none" /> Zeilenumbruch.
</p>
<ul>
<li>Listen Element</li>
<li>Entity Element &abc; </li>
</ul>
<h1>Überschrift B</h1>
<p>Text eines zweiten Paragraphen mit
<strong>Hervorhebung</strong>
ohne Zeilenumbruch.
</p>
<h1 count="10">Überschrift C</h1>
<p>Text eines dritten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
<h1>Überschrift D</h1>
<p>Text eines vierten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
</body>
</html>

```

DOMUtil:

```

import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element;

public class DOMUtil {

    private static int inc = 0;

    public static void addIdent(Node node, String prefix) {
        switch ( node.getNodeType() ) {
            case Node.ELEMENT_NODE:
                addElementIdent( (Element)node, prefix );
                break;
            case Node.DOCUMENT_NODE:
                addElementIdent( ((Document)node).getDocumentElement(), prefix );
                break;
            default: ;
        }
    }

    public static void addElementIdent(Element el, String prefix) {
        el.setAttribute("id", "#" + prefix + inc);
        el.setAttribute("name", "#" + prefix + inc++);
        NodeList nl = el.getChildNodes();
        if ( ( nl == null ) || ( nl.getLength() == 0 ) ) {
            return;
        }
        for (int i = 0; i < nl.getLength(); i++) {
            Node c = nl.item(i);
            addIdent(c, prefix);
        }
    }
}

```

```
}
    }
}
```

Anwendung:

```
// ... wie oben
DOMUtil.addIdent(doc, "hk");

sc.println("document:\n");
dw.printNode(doc);
dw.flush();
sc.println();
```

Ausgabe:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "DTD/xhtml1-transitional.dtd" >
<html id="#hk0" name="#hk0" xmlns="http://www.w3.org/1999/xhtml">
<head id="#hk1" name="#hk1">
<title id="#hk2" name="#hk2">Titel</title>
<?php $a=9;
    function yyy(a) {
        if ( a > 2 ) {
            print("<p>nonsense</p>");
        }
    }
?>
<!-- funktions definitionen -->
<script id="#hk3" language="JavaScript" name="#hk3" type="text/javascript">
<![CDATA[ ]]>
</script>
</head>
<body bgcolor="white" id="#hk4" name="#hk4">
<h1 id="#hk5" name="#hk5">Überschrift A</h1>
<p align="center" id="#hk6" name="#hk6">Paragraph mit <em id="#hk7" name="#hk7">Hervorhebung</em>
und einem <br clear="none" id="#hk8" name="#hk8" /> Zeilenumbruch.
</p>
<ul id="#hk9" name="#hk9">
<li id="#hk10" name="#hk10">Listen Element</li>
<li id="#hk11" name="#hk11">Entity Element &abc; </li>
</ul>
<h1 id="#hk12" name="#hk12">Überschrift B</h1>
<p id="#hk13" name="#hk13">Text eines zweiten Paragraphen mit
<strong id="#hk14" name="#hk14">Hervorhebung</strong>
ohne Zeilenumbruch.
</p>
<h1 count="10" id="#hk15" name="#hk15">Überschrift C</h1>
<p id="#hk16" name="#hk16">Text eines dritten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
<h1 id="#hk17" name="#hk17">Überschrift D</h1>
<p id="#hk18" name="#hk18">Text eines vierten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
</body>
</html>
```

19.3. SAX Parser

- Apache Xerces: org.apache.xerces.parser.SAXParser
- im JDK ab 1.4 als: javax.xml.parsers.SAXParser
und javax.xml.parsers.SAXParserFactory

SAXParserFactory

`javax.xml.parsers.SAXParserFactory` dient der Erzeugung eines SAX Parsers `javax.xml.parsers.SAXParser`.

```
protected SAXParserFactory()
public static newInstance()

public SAXParser newSAXParser()

boolean isValidating()
void setValidating(boolean)

boolean isExpandEntityReferences()
void setExpandEntityReferences(boolean)

boolean isNamespaceAware()
void setNamespaceAware(boolean)
```

SAXParser

`javax.xml.parsers.SAXParser` dient dem Parsen, d.h. dem Einlesen und Verarbeiten der gefundenen XML Konstrukte.

```
protected SAXParser()

public void parse(File f, DefaultHandler dh)
public void parse(InputStream is, DefaultHandler dh)
public void parse(InputSource is, DefaultHandler dh)
public void parse(String uri, DefaultHandler dh)

public boolean isValidating()

public abstract void setProperty(String name, Object value)
public abstract Object getProperty(String name)
```

SAXLeser:

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class SAXLeser {

    public static void main(String[] args) {

        SAXParserFactory pfac = SAXParserFactory.newInstance();
        SAXParser parser = null;

        try {
            pfac.setValidating(true);
            pfac.setNamespaceAware(true);
            parser = pfac.newSAXParser();
        }
        catch (ParserConfigurationException e) {
            e.printStackTrace();
        }
        catch (SAXException e) {
            e.printStackTrace();
        }

        Screen sc = new Screen();
        String uri = "datei.xhtml";
        DefaultHandler dh = new SAXSchreiber(sc);

        try {
            parser.parse(uri,dh);
        }
```

```
    }  
    catch (SAXException e) {  
        e.printStackTrace();  
    }  
    catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

DefaultHandler aus `org.xml.sax.helpers`,
implementiert die Interfaces `ContentHandler`, `DTDHandler`, `EntityResolver` und `ErrorHandler`:

```
public void startDocument()  
public void endDocument()  
  
public void startElement(String nameuri, String local, String quali,  
                        Attributes attrs)  
public void endElement(String nameuri, String local, String quali)  
  
public void characters(char ch[], int start, int length)  
public void ignorableWhitespace(char ch[], int start, int length)  
  
public void processingInstruction(String target, String data)  
  
public void warning(SAXParseException ex)  
public void error(SAXParseException ex)  
public void fatalError(SAXParseException ex)  
  
public InputSource resolveEntity(String publicId,  
                                String systemId)  
  
public void notationDecl(String name, String publicId,  
                        String systemId)
```

SAXSchreiber:

```
import java.io.IOException;  
import java.io.PrintWriter;  
import org.xml.sax.SAXException;  
import org.xml.sax.SAXParseException;  
import org.xml.sax.Attributes;  
import org.xml.sax.InputSource;  
import org.xml.sax.helpers.DefaultHandler;  
  
public class SAXSchreiber extends DefaultHandler {  
    PrintWriter pw = null;  
  
    public SAXSchreiber() {  
        pw = new Screen();  
    }  
  
    public SAXSchreiber(PrintWriter p) {  
        pw = p;  
    }  
  
    public void startDocument() {  
        pw.println("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>");  
        pw.flush();  
    }  
  
    public void endDocument() {  
        pw.println();  
        pw.flush();  
    }  
}
```

```

public void startElement(String nameuri, String local, String quali,
                        Attributes attrs) {
    pw.print('<');
    pw.print(local);
    if (attrs != null) {
        int len = attrs.getLength();
        for (int i = 0; i < len; i++) {
            pw.print(' ');
            pw.print(attrs.getQName(i));
            pw.print("=\");
            pw.print(attrs.getValue(i));
            pw.print("\"");
        }
    }
    pw.print('>');
    pw.flush();
}

public void endElement(String nameuri, String local, String quali) {
    pw.print("</");
    pw.print(local);
    pw.print('>');
    pw.flush();
}

public void characters(char ch[], int start, int length) {
    pw.print( new String(ch, start, length) );
    pw.flush();
}

public void ignorableWhitespace(char ch[], int start, int length) {
    characters(ch, start, length);
}

public void processingInstruction(String target, String data) {
    pw.print("<?");
    pw.print(target);
    if (data != null && data.length() > 0) {
        pw.print(' ');
        pw.print(data);
    }
    pw.print(">");
    pw.flush();
}

public void warning(SAXParseException ex) {
    pw.println("\n<!-- [Warning] "+
               ex.getSystemId()+": line "+
               ex.getLineNumber()+", column "+
               ex.getColumnNumber()+":\n"+
               ex.getMessage() + " -->");
}

public void error(SAXParseException ex) {
    pw.println("\n<!-- [Error] "+
               ex.getSystemId()+": line "+
               ex.getLineNumber()+", column "+
               ex.getColumnNumber()+":\n"+
               ex.getMessage() + " -->");
}

public void fatalError(SAXParseException ex) {
    pw.println("\n<!-- [Fatal Error] "+
               ex.getSystemId()+": line "+
               ex.getLineNumber()+", column "+
               ex.getColumnNumber()+":\n"+
               ex.getMessage() + " -->");
}

public InputSource resolveEntity(String publicId,

```

```

        String systemId) {
    pw.println("\n<!-- [resolveEntity] publicId: " +
        publicId + ", systemId: " +
        systemId + " -->");
    boolean dtd = false;
    boolean pid = false;
    boolean sid = false;
    if ( ( publicId != null ) || ( systemId != null ) ) {
        pid = pid || ( publicId.indexOf("DTD") >= 0 );
        pid = pid || ( publicId.indexOf("dtd") >= 0 );
        sid = sid || ( systemId.indexOf("DTD") >= 0 );
        sid = sid || ( systemId.indexOf("dtd") >= 0 );
    }
    if (pid) {
        pw.print("<!DOCTYPE ????" );
        pw.print("PUBLIC \"" + publicId + "\"");
        if (sid) {
            pw.print(" \"" + systemId + "\"");
            pw.println(">");
        }
    } else if (sid) {
        pw.print("<!DOCTYPE ????" );
        pw.print("SYSTEM \"" + systemId + "\"");
        pw.println(">");
    }
    return null;
}

public void notationDecl(String name, String publicId,
                        String systemId) {
    pw.println("\n[notationDeclaration] name: " + name + " publicId: " +
        publicId + ", systemId: " +
        systemId );
}
}

```

Attributes aus org.xml.sax,
 AttributesImpl aus org.xml.sax.helpers:
 interface Attributes

```

int getLength()

int getIndex(String qName)
int getIndex(String uri, String localName)

String getQName(int index)
String getLocalName(int index)
String getURI(int index)

String getValue(int index)
String getValue(String qName)
String getValue(String uri, String localName)

String getType(int index)
String getType(String qName)
String getType(String uri, String localName)

```

class AttributesImpl plus

```

AttributesImpl()
AttributesImpl(Attributes atts)

void clear()
void removeAttribute(int index)

void addAttribute(String uri, String localName, String qName,
                 String type, String value)
void addAttribute(int index,
                 String uri, String localName, String qName,
                 String type, String value)

```

```
void setAttributes(Attributes atts)
void setLocalName(int index, String localName)
void setQName(int index, String qName)
void setType(int index, String type)
void setURI(int index, String uri)
void setValue(int index, String value)
```

SAXUtil: Nummerieren der h1-Überschriften

```
import java.io.IOException;
import java.io.PrintWriter;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.AttributesImpl;

public class SAXUtil extends SAXSchreiber {

    int chapter = 0;
    boolean insertCounter = false;

    public SAXUtil() {
        pw = new Screen();
    }

    public SAXUtil(PrintWriter p) {
        pw = p;
    }

    public void startDocument() {
        chapter = 0;
        super.startDocument();
    }

    public void startElement(String nameuri, String local, String quali,
                             Attributes atts) {
        if ( !quali.equals("h1") ) {
            super.startElement(nameuri, local, quali, atts);
            pw.flush();
            return;
        }
        Attributes at = atts;
        String cVal = null;
        if ( at != null ) {
            cVal = at.getValue("count");
            if ( cVal == null ) {
                cVal = "" + (++chapter);
                AttributesImpl ai = new AttributesImpl(at);
                ai.addAttribute("", "", "count", "", cVal);
                at = (Attributes) ai;
            } else {
                chapter = Integer.parseInt(cVal);
            }
        } else {
            AttributesImpl ai = new AttributesImpl();
            cVal = "" + (++chapter);
            ai.addAttribute("", "", "count", "", cVal);
            at = (Attributes) ai;
        }
        insertCounter = true;
        super.startElement(nameuri, local, quali, at);
    }

    public void characters(char ch[], int start, int length) {
        String s = new String(ch, start, length);
    }
}
```

```

        if ( insertCounter ) {
            s = "Kapitel " + chapter + ". " + s;
            insertCounter = false;
        }
        pw.print( s );
        pw.flush();
    }

    public InputSource resolveEntity(String publicId,
                                    String systemId) {
        return null;
    }

    public void notationDecl(String name, String publicId,
                             String systemId) {
    }
}

```

Ausgabe für 'datei.xml' von oben:

```

<?xml version="1.0" encoding="iso-8859-1"?>

<!-- [resolveEntity] publicId: -//W3C//DTD XHTML 1.0 Transitional//EN, systemId: DTD/xhtml1-transitional.dtd -->
<!DOCTYPE ??? PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd" >

<!-- [resolveEntity] publicId: -//W3C//ENTITIES Latin 1 for XHTML//EN, systemId: xhtml-lat1.ent -->
<!-- [resolveEntity] publicId: -//W3C//ENTITIES Symbols for XHTML//EN, systemId: xhtml-symbol.ent -->
<!-- [resolveEntity] publicId: -//W3C//ENTITIES Special for XHTML//EN, systemId: xhtml-special.ent -->
<html>
<head>
<title>Titel</title>
<?php $a=9;
    function yyy(a) {
        if ( a > 2 ) {
            print("<p>nonsense</p>");
        }
    }
?>
<script type="text/javascript" language="JavaScript">

    function xxx(a) {
        if ( a > 2 ) {
            document.write("<p>nonsense</p>");
        }
    }
</script>
</head>
<body bgcolor="white">
<h1>Überschrift A</h1>
<p align="center">Paragraph mit <em>Hervorhebung</em>
und einem <br clear="none"></br> Zeilenumbruch.
</p>
<ul>
<li>Listen Element</li>
<li>Entity Element &abc; </li>
</ul>
<h1>Überschrift B</h1>
<p>Text eines zweiten Paragraphen mit
<strong>Hervorhebung</strong>
ohne Zeilenumbruch.
</p>

<!-- [Error] datei.xml: line 41, column 16:
Attribute "count" must be declared for element type "h1". -->
<h1 count="10">Überschrift C</h1>
<p>Text eines dritten Paragraphen ohne

```

```
Hervorhebungen und ohne Zeilenumbruch.  
</p>  
<h1>Überschrift D</h1>  
<p>Text eines vierten Paragraphen ohne  
Hervorhebungen und ohne Zeilenumbruch.  
</p>  
</body>  
</html>
```

SAXNummer:

```
import javax.xml.parsers.SAXParserFactory;  
import javax.xml.parsers.SAXParser;  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.IOException;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
public class SAXNummer {  
    public static void main(String[] args) {  
        SAXParserFactory pfac = SAXParserFactory.newInstance();  
        SAXParser parser = null;  
  
        try {  
            // pfac.setValidating(true);  
            pfac.setNamespaceAware(true);  
            parser = pfac.newSAXParser();  
        } catch (ParserConfigurationException e) {  
            e.printStackTrace();  
        }  
        catch (SAXException e) {  
            e.printStackTrace();  
        }  
  
        Screen sc = new Screen();  
        String uri = "datei.xhtml";  
        DefaultHandler dh = new SAXUtil(sc);  
  
        try {  
            parser.parse(uri,dh);  
        } catch (SAXException e) {  
            e.printStackTrace();  
        }  
        catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Ausgabe für 'datei.xhtml' von oben:

```
<?xml version="1.0" encoding="iso-8859-1"?>  
<html>  
  <head>  
    <title>Titel</title>  
    <?php $a=9;  
      function yyy(a) {  
        if ( a > 2 ) {  
          print("<p>nonsense</p>");  
        }  
      }  
    ?>  
  
    <script type="text/javascript" language="JavaScript">  
      function xxx(a) {
```

```
        if ( a > 2 ) {
            document.write("<p>nonsense</p>");
        }
    }
</script>
</head>
<body bgcolor="white">
<h1 count="1">Kapitel 1. Überschrift A</h1>
<p align="center">Paragraph mit <em>Hervorhebung</em>
und einem <br clear="none"></br> Zeilenumbruch.
</p>
<ul>
<li>Listen Element</li>
<li>Entity Element &abc; </li>
</ul>
<h1 count="2">Kapitel 2. Überschrift B</h1>
<p>Text eines zweiten Paragraphen mit
<strong>Hervorhebung</strong>
ohne Zeilenumbruch.
</p>
<h1 count="10">Kapitel 10. Überschrift C</h1>
<p>Text eines dritten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
<h1 count="11">Kapitel 11. Überschrift D</h1>
<p>Text eines vierten Paragraphen ohne
Hervorhebungen und ohne Zeilenumbruch.
</p>
</body>
</html>
```

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Wed May 26 23:06:38 CEST 2004

20. Metadaten: DC und RDF

- Einleitung
 - Dublin Core (DC)
 - Resource Description Framework (RDF)
 - RDF Sprachkonstrukte
 - Zusammenfassung und Ausblick
-

20.1. Einleitung

- Metadaten: Daten über Daten
Informationen über Informationen
 - Klassifikation der Daten-Inhalte
 - Problem: viele Bereiche mit unterschiedlicher Klassifikation
Pizza-Service, Bücher, Autos, Telefonbuch, Dienstleistungen
 - Problem: kein anerkanntes Schema zur Klassifikation
 - Bibliothekare haben viele Schemata entwickelt
 - Dublin Core, 1995 definiert
Dublin in USA
 - Klassifikation durch Menschen: z.B. Yahoo
Klassifikation durch Maschinen: Volltextsuche
 - Resource Description Framework (RDF)
Rahmen für unterschiedliche Metadaten-Systeme
 - RDF ist kompatibel zu XML
 - (Fern-) Ziel: Semantic Web
-

20.2. Dublin Core (DC)

Ziele für den Entwurf

- Einfachheit:
durch Nicht-Experten benutzbar
- Semantische Kompatibilität:
über Fachgrenzen hinweg benutzbar
- Internationaler Konsens:
von Leuten aus über 30 Ländern erarbeitet
- Erweiterbarkeit:
offen für feinere Untergliederung der Metadaten
- Anwendbarkeit im Web:
kompatibel mit RDF

Die Bedeutung der Elemente ist wiederum durch die Spezifikation von ISO/ITEC 11179 Attributen festgelegt.

Syntax

- unterscheidet "Eigenschaften" und deren "Werte/Inhalte"
- `DC.Eigenschaft="dc-wert"`
- alle Eigenschaften sind optional und wiederholbar
- die Reihenfolge der Angabe der Eigenschaften ist beliebig
- Auflistung:
`DC.Eigenschaft="dc-wert1; dc-wert2; dc-wert3"`
- Hierarchie:
`DC.Eigenschaft="dc-wert-1. dc-wert-2. dc-wert-3"`
- Einbettung in HTML mit meta-Element
`<meta name="DC.Eigenschaft" content="dc-wert">`
Wiederholung durch mehrere meta-Elemente
- Einbettung in XHTML und XML Dokumente mit RDF
`<dc:Eigenschaft> dc-wert </dc:Eigenschaft>`
als Attribut `dc:Eigenschaft="dc-wert"`
Wiederholung durch mehrere dc:Eigenschafts-Elemente oder RDF Container

Die Core Elemente

Inhalt	Intellektuelle Zugehörigkeit	Status
Coverage	Contributor	Date
Description	Creator	Format
Type	Publisher	Identifier
Relation	Rights	Language
Source		
Subject		
Title		

Titel

Label: Title

Definition: Name der Quelle.

Beispiel:

`DC.Title="A Pilot's Guide to Aircraft Insurance"`

`DC.Title="The Sound of Music"`

`DC.Title="Green on Greens"`

`DC.Title="AOPA's Tips on Buying Used Aircraft"`

Autor oder Erschaffer

Label: Creator

Definition: Körperschaft/Person, die für die Quelle inhaltlich verantwortlich ist.

Beispiel:

`DC.Creator="Duncan, Phyllis-Anne"`

`DC.Creator="Melendez Santiago; Maria Luz"`

DC.Creator="Maimonides"

aber:

DC.Creator="Park Sung Hee"

Im Falle von Organisationen bei denen eine klare Hierarchie vorhanden ist, listen sie die Teile dieser Hierarchie von Grösstem zum Kleinstem, getrennt durch Punkte.

Beispiel:

DC.Creator="United States. Internal Revenue Service"

DC.Creator="Elvis Presley Fan Club"

DC.Creator="Federal Aviation

Administration. Aviation Safety Program."

nicht:

DC.Creator="Aviation Safety Program of the Federal Aviation Administration"

DC.Creator="Art Institute of Chicago"

DC.Creator="Association of the Bar of the City of New York"

DC.Creator="Baltimore County Medical Society"

Thema und Schlüsselwörter

Label: Subject

Definition: Thema mit dem sich die Quelle beschäftigt.

Beispiel:

DC.Subject="Aircraft leasing and renting"

DC.Subject="Dogs"

DC.Subject="Olympic skiing"

DC.Subject="Street, Picabo"

Beschreibung

Label: Description

Definition: Überblick über den Inhalt der Quelle (Abstract, Inhaltsverzeichnis).

Beispiel:

DC.Description="Illustrated guide to airport markings and lighting signals, with particular reference to SMGCS (Surface Movement Guidance and Control System) for airports with low visibility conditions"

Herausgeber

Label: Publisher

Definition: Körperschaft/Person, die für die Verfügbarkeit der Quelle verantwortlich ist.

Beispiel:

DC.Publisher="Moguls Anonymous"

DC.Publisher="University of Miami. Dept. of Economics"

DC.Publisher="Free Software Foundation"

Datum

Label: Date, Format: YYYY-MM-DD oder YYYY-MM oder YYYY

Definition: Datum der Erstellung oder Veröffentlichung der Quelle.

Beispiel:

DC.Date="1998-02-16"

DC.Date="1998-02"

DC.Date="1998"

Art oder Type der Quelle

Label: Type

Definition: Art oder Genre der Quelle.

Minimale Liste, die für DC empfohlen ist:

- text
- image
- sound
- data
- software
- interactive
- physical object

Beispiel:

DC.Type="image"

DC.Type="sound"

DC.Type="text"

DC.Type="image"

Multimedia educational program with interactive assignments:

DC.Type="text" DC.Type="image"

DC.Type="software" DC.Type="interactive"

Format

Label: Format, MIME Type

Definition: physikalische oder digitale Manifestation der Quelle (Datenformat, Systemvoraussetzungen).

Beispiel:

DC.Format="image/gif"

Identifikation der Quelle

Label: Identifier

Definition: eindeutige Referenz der Quelle (URL, ISBN, DOI).

Beispiel:

DC.Identifier="http://purl.oclc.org/metadata/dublin_core/"

DC.Identifier="0385424728" [ISBN]

DC.Identifier="H-A-X 5690B" [publisher number]

Ursprung

Label: Source

Definition: Referenz zum Ursprung der Quelle.

Beispiel:

DC.Source="RC607.A26W574 1996"

wobei "RC607.A26W574 1996" z.B. eine Bezeichnung des gedruckten Werkes ist

Sprache

Label: Language

Definition: Sprache(n) des Inhalts der Quelle.

Beispiel:

DC.Language="en" DC.Language="fr"

oder

DC.Language="en;fr"

oder

DC.Language="Primarily English, with some abstracts
also in French."

DC.Language="en-US"

Beziehungen zu anderen Quellen

Label: Relation

Definition: Referenz auf verwandte Quellen.

Eine Liste von Beziehungstypen:

- IsPartOf
- HasPart

- IsVersionOf
- HasVersion
- IsFormatOf
- HasFormat
- References
- IsReferencedBy
- IsBasedOn
- IsBasisFor
- Requires
- IsRequiredBy

DC.Title="the present resource"

DC.Relation="relationship-type [space] unique identifier
for the related resource"

wobei "relationship-type" aus obiger Liste stammt

Note: In the case where the DC metadata is embedded in the present resource, the value for Identifier is implied (i.e. the present resource). In qualified DC the two components given in Relation here will be structured using sub-elements for easier automated processing.

Beispiel:

DC.Title="Reading Turgenev"
DC.Relation="IsPartOf TwoLives"

eine Sammlung von zwei Novellen, von denen eine "Reading Turgenev" ist

Rechtmanagement

Label: Rights

Definition: Informationen über die Urheberrechte an der Quelle.

Beispiel:

DC.Rights="http://cs-tr.cs.cornell.edu/Dienst/Repository/2.0/Terms"

Themen-Abdeckung

Label: Coverage

Beispiel:

DC.Coverage="1995-1996"

DC.Coverage="Boston, MA"

oder

DC.Coverage="17th century"

DC.Coverage="Upstate New York"

Sonstige Beitragende

Label: Contributor

Definition: Sonstige Beitragende zur Quelle.

Beispiele

Diese Seite mit dem standard HTML Meta Element

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="GENERATOR" content="Emacs" />
<meta name="AUTHOR" content="Heinz Kredel" />
<meta name="CREATED" content="20001208;" />
```

Diese Seite mit dem HTML Meta Element und DC

```
<meta name="DC.Title" content="DC and RDF" />
<meta name="DC.Creator" content="Kredel, Heinz" />
<meta name="DC.Subject" content="Metadata Systems DC and RDF" />
<meta name="DC.Description" content="Introduction to the DC and RDF Metadata Systems" />
<meta name="DC.Publisher" content="University of Mannheim" />
<meta name="DC.Date" content="2000-12-08" />

<meta name="DC.Type" content="text; image" />
<meta name="DC.Format" content="text/xhtml; image/gif" />
<meta name="DC.Identifier" content="http://krum.rz.uni-mannheim.de/inet-2004/sess-308.html" />
<meta name="DC.Language" content="de; en" />
<meta name="DC.Relation" content="IsPartOf http://krum.rz.uni-mannheim.de/inet-2004/" />
```

20.3. Resource Description Framework (RDF)



- Ziel: automatisierte Auswertung und Bearbeitung von Metadaten
- Auffinden von Quellen (im Web)
- Katalogisierung (von Web Inhalten)
- Bewertung (Rating) (von Web Inhalten)
- Baustein des *Web of Trust*
- Nachfolger von PICS (Platform for Internet Content Selection)
- W3C Recommendation, 22 February 1999

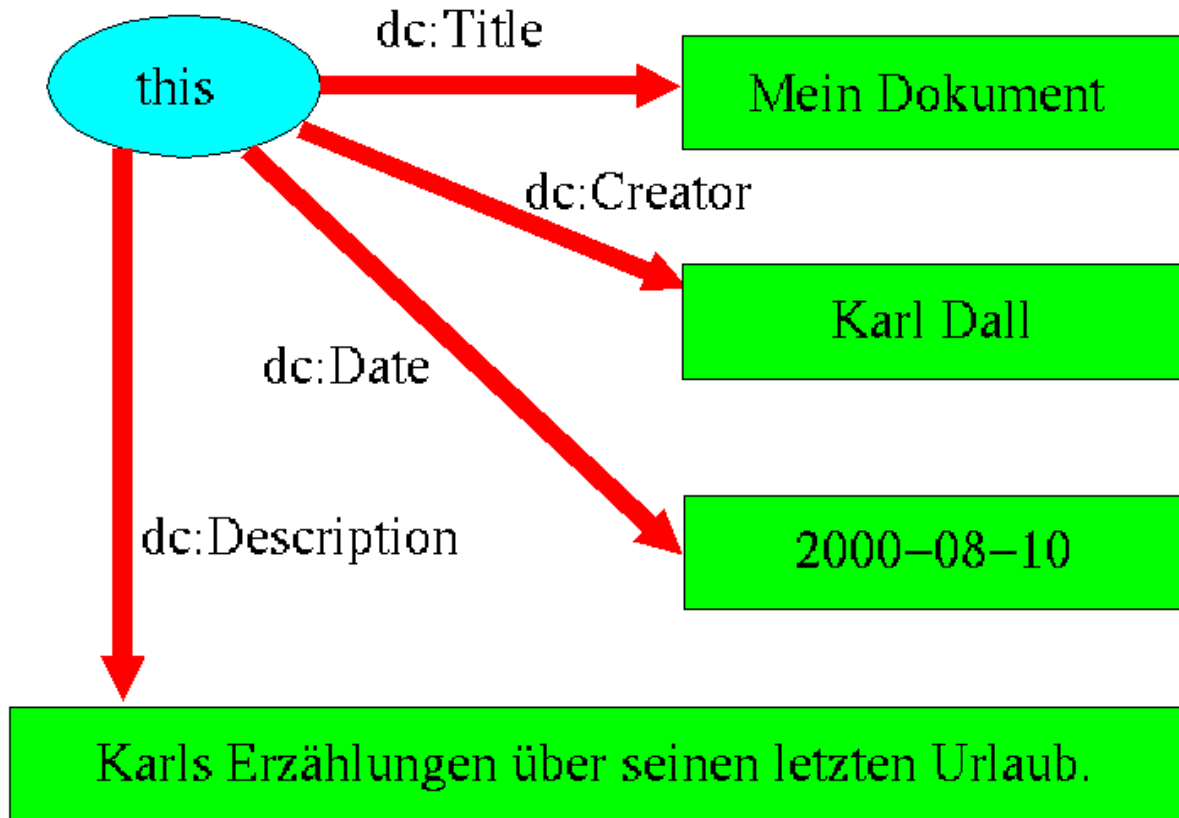
Features von RDF

- Interoperabilität zwischen Metadaten-Systemen
- durch Computer verwertbare Metadaten
- Präzision für Metadaten
genauere Markierungen statt einfachen Volltexten
- Offenheit für zukünftige Erweiterungen

Beispiel für RDF mit DC innerhalb einer (X-)HTML Seite

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description rdf:about="" [d.h. für dieses Dokument]
```

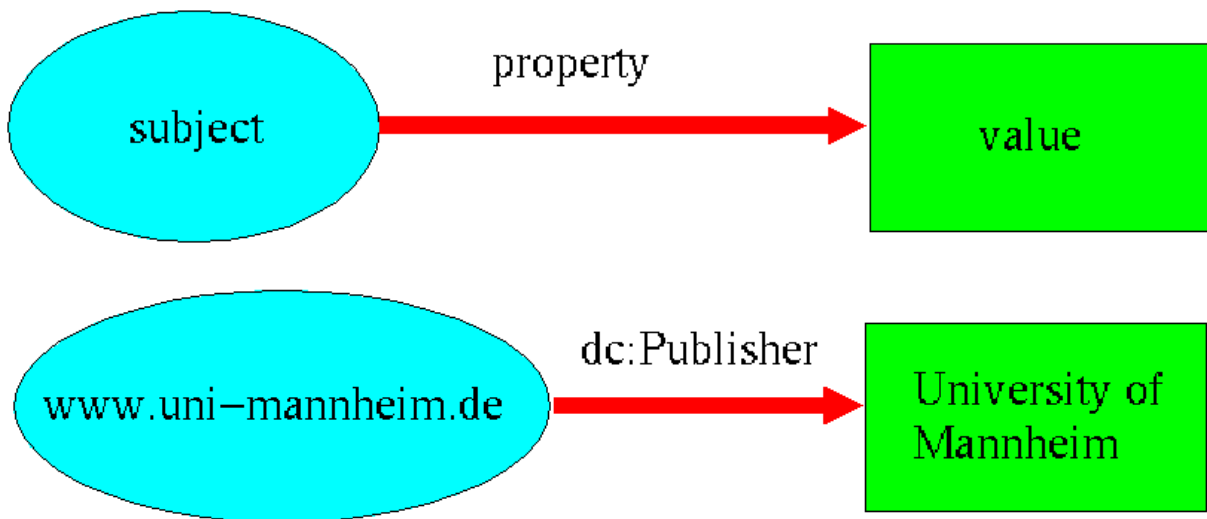
```
dc:creator="Karl Dall"  
dc:title="Mein Dokument"  
dc:description="Karls Erzählungen über seinen letzten Urlaub."  
dc:date="2000-08-10" />  
</rdf:RDF>
```



RDF ist ein Framework für Metadaten
d.h. Metasystem für Metadaten
d.h. Daten über (Daten über Daten)

Grundlegendes Datenmodell von RDF:

1. **Resource:**
alles was einen URI (plus Fragment-Identifier) haben kann, d.h. alle Web-Dokumente und per Xpointer selektierbare Teile, aber z.B. auch realexistierende Bücher
2. **Property**
ein Aspekt, eine Eigenschaft, ein Attribut einer Resource, kann selbst Resource sein, kann einen Namen haben, z.B. Autor oder Titel, kann wieder eigene Properties haben
3. **Statement**
Beziehung zwischen einer Resource, einer Property und einem Wert, ein Wert kann eine Zeichenkette oder wieder eine Resource sein
definiert Trippel: (*subject*, *predicate*, *object*)
(eine Resource, eine Property, Wert)



Charakteristika

- **Unabhängigkeit:**
da eine Property eine Resource ist, kann jeder seine eigenen erfinden
- **Austauschbarkeit**
da RDF auf XML basiert, kann es leicht kommuniziert und ausgetauscht werden
- **Skalierbarkeit**
da ein Statement nur aus den drei Teilen (Resource, Property, Wert) besteht, können diese in grossen Mengen maschinell verarbeitet werden
- **Properties sind Ressourcen**
da Properties selbst wieder Ressourcen sind, können sie eigene Properties haben und diese können per RDF automatisch verarbeitet werden
- **Werte können Ressourcen sein**
da Werte selbst wieder Ressourcen sein können, können sie auch wieder eigene Properties haben
- **Statements können Ressourcen sein**
da Statements selbst wieder Ressourcen sein können, können sie auch wieder eigene Properties haben

Was RDF nicht bietet

- definiert selbst kein **Vokabular** (wie z.B. Dublin Core) für Metadaten
- RDF ist nicht selbst durch eine XML DTD definiert sondern direkt durch EBNF (Extended Backus-Naur Form)
- RDF Konzept ist unabhängig von XML kompatibler Syntax
- Reihenfolge der Definitionen ist nicht signifikant
- Datenstrukturen für Dokumente einer XML DTD sind erheblich komplexer, z.B. wegen Mischung von #PCDATA und Elementen

RDF Sprachkonstrukte

zwei syntaktische Varianten

- **Serialization Syntax**
kompatibel zu XML DTDs, ungeeignet zur Einbettung in HTML
Properties als XML-Elemente, Problem: Element Inhalte

- Abbreviated Syntax
(teilweise) nicht kompatibel zu XML DTDs, geeignet zur Einbettung in HTML
Properties als XML-Attribute, Problem: mehrfache Attribute

RDF Basis Element: Description

(mehrere) **Statements** können durch das Description-Element definiert werden

- `<rdf:Description about="URI#Xpointer">`
PropElem*
`</rdf:Description>`
Bezeichnung eines (externen) **Subjekts**
falls "URI#Xpointer" = "" Definition des aktuellen Dokuments als Subjekt
- `<rdf:Description ID="identifizier">`
PropElem*
`</rdf:Description>`
Definition dieser Statements als **Subjekt** der Resource mit Bezeichnung "identifizier"
- `<rdf:Description>`
PropElem*
`</rdf:Description>`
Anonyme **Subjekt** Definition diese(s/r) Statements

RDF Property Elemente

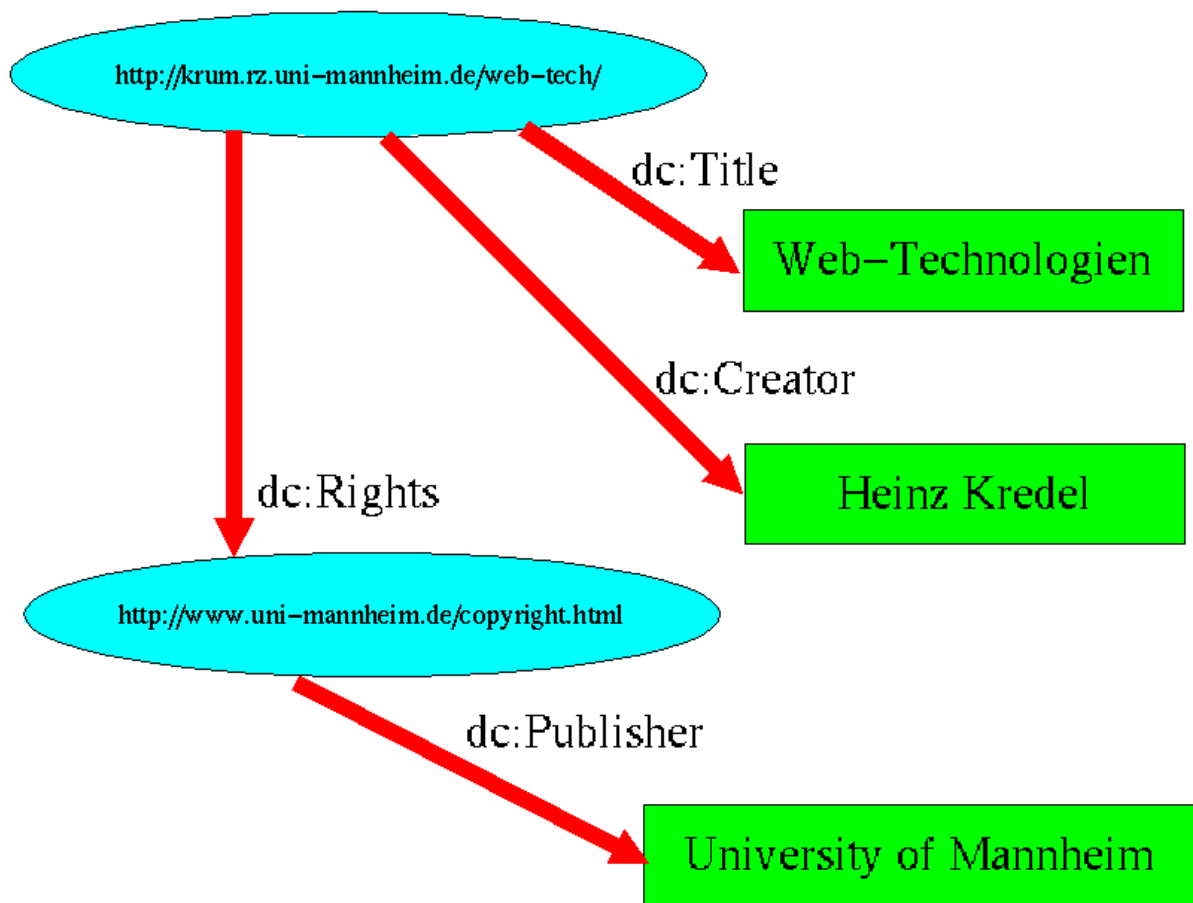
PropElem* definiert eine Folge von **Properties** als XML-Elemente

- `<propName resource="URI#Xpointer" />`
Definition der Property "propName" als (externe) **Resource**
- `<propName> wert </propName>`
Definition von "wert" als **Wert** der Property "propName"

wobei propName ein mit XML-Namespace qualifizierter Name sein kann

Beispiel

```
<rdf:Description about="http://krum.rz.uni-mannheim.de/web-tech/">
  <dc:creator>Heinz Kredel</dc:creator>
  <dc:title>Web-Technologien</dc:title>
  <dc:rights rdf:resource="http://www.uni-mannheim.de/copyright.html" />
</rdf:Description>
```



RDF Property Attribute

für die Abbreviated Syntax können statt der Property-Elemente Property-Attribute verwendet werden

- `<rdf:Description ..subj.. PropAttr* />`
Definition des **Subjekts** ..subj.. wie in der Serialization Syntax

PropAttr* definiert eine Folge von **Properties** als XML-Attribute

- PropAttr ist `propName="URI#Xpointer"`
Definition der Property "propName" als (externe) **Resource**
- PropAttr ist `propName="wert"`
Definition von "wert" als **Wert** der Property "propName"

wobei propName auch wieder ein mit XML-Namespaces qualifizierter Name sein kann

RDF Hilfselemente

- Container: Sequence, Bag, Alternative
- `parseType="Literal"`

Transport von RDF Descriptions

1. Eingebettet in die Resource
wie in HTML oder XHTML
2. Extern zur Resource, aber automatisch mitgeliefert
3. Extern zur Resource, Lieferung nur per expliziter Aufforderung
4. Umschliessung der Resource
d.h. die Resource ist eingebettet in die RDF Description

bei 2 und 3 sollte folgende Syntax verwendet werden

```
<link rel="meta" href="mydocMetadata.dc.rdf">
```

Beispiele

Diese Seite mit RDF Einbettung in XHTML

in Abbreviated Syntax

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/" >
<rdf:Description about=""
  dc:Title      ="DC and RDF"
  dc:Creator    ="Kredel, Heinz"
  dc:Subject     ="Metadata Systems DC and RDF"
  dc:Description ="Introduction to the DC and RDF Metadata Systems"
  dc:Publisher   ="University of Mannheim"
  dc:Date        ="2000-12-08"

  dc:Type        ="text; image"
  dc:Format       ="text/xhtml; image/gif"
  dc:Identifier   ="http://krum.rz.uni-mannheim.de/inet-2004/sess-308.html"
  dc:Language     ="de; en"
  dc:Relation      ="IsPartOf http://krum.rz.uni-mannheim.de/inet-2004/"
/>
</rdf:RDF>
```

in Serialization Syntax

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/" >
<rdf:Description about="" >
  <dc:Title      >DC and RDF</dc:Title >
  <dc:Creator    >Kredel, Heinz</dc:Creator >
  <dc:Subject     >Metadata Systems DC and RDF</dc:Subject >
  <dc:Description >Introduction to the DC and RDF Metadata Systems</dc:Description >
  <dc:Publisher   >University of Mannheim</dc:Publisher >
  <dc:Date        >2000-12-08</dc:Date >

  <dc:Type        >text; image</dc:Type >
  <dc:Format       >text/xhtml; image/gif</dc:Format >
  <dc:Identifier   >http://krum.rz.uni-mannheim.de/inet-2004/sess-308.html</dc:Identifier >
  <dc:Language     >de; en</dc:Language >
  <dc:Relation      >IsPartOf http://krum.rz.uni-mannheim.de/inet-2004/</dc:Relation >
</rdf:Description>
</rdf:RDF>
```

in externer Datei [sess-308.html.rdf](#) oder [sess-308.html.rdf.txt](#),
Zugriff mit:

```
<link rel="meta" href="sess-308.html.rdf" />
```

Mozilla verwendet RDF

Der Mozilla Browser verwendet RDF für einige seiner Konfigurationsdateien. z.B. für die Zuordnung von Mime Typen zu Anwendungen (mimeTypes.rdf):

```
<?xml version="1.0"?>
<RDF:RDF xmlns:NC="http://home.netscape.com/NC-rdf#"
  xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <RDF:Description about="urn:mimetype:externalApplication:application/pdf"
    NC:path="acroread"
    NC:prettyName="acroread" />
  <RDF:Description about="urn:mimetype:application/pdf"
    NC:value="application/pdf"
    NC:description=""
    NC:fileExtensions="pdf"
    NC:editable="true">
    <NC:handlerProp resource="urn:mimetype:handler:application/pdf"/>
  </RDF:Description>
  <RDF:Description about="urn:mimetypes">
    <NC:MIME-types resource="urn:mimetypes:root"/>
  </RDF:Description>
  <RDF:Description about="urn:mimetype:handler:application/pdf"
    NC:saveToDisk="false"
    NC:handleInternal="false"
    NC:alwaysAsk="true">
    <NC:externalApplication
      resource="urn:mimetype:externalApplication:application/pdf"/>
  </RDF:Description>
  <RDF:Seq about="urn:mimetypes:root">
    <RDF:li resource="urn:mimetype:application/pdf"/>
  </RDF:Seq>
</RDF:RDF>
```

RDF Site Summary: RSS

Zum Zusammenfassen und Verbreiten von Nachrichtenströmen (news feed syndication) gibt es das RDF basierte RSS System (manchmal auch unter dem Namen Rich Site Summary).

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://purl.org/rss/1.0/"
  xmlns:chump="http://usefulinc.com/ns/chump#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <channel rdf:about="http://rdfig.xmlhack.com/index.rss">
    <link>http://rdfig.xmlhack.com/</link>
    <title>RDF Interest Group Scratchpad</title>
    <description> -- last modified 2003-06-24 17:03</description>
    <items>
      <rdf:Seq>
        <rdf:li
          rdf:resource="http://bitsko.slc.ut.us/blog/2003/06/24/foaf-check"/>
        <rdf:li rdf:resource="http://rdfweb.org/" />
        <rdf:li rdf:resource="http://www.w3.org/2003/03/semantic-tour"/>
        <rdf:li rdf:resource="http://www.w3.org/News/2003#item105"/>
      </rdf:Seq>
    </items>
  </channel>
  <item rdf:about="http://bitsko.slc.ut.us/blog/2003/06/24/foaf-check">
    <link>http://bitsko.slc.ut.us/blog/2003/06/24/foaf-check</link>
    <chump:contributor>
      <foaf:Person>
        <foaf:nick>bitsko</foaf:nick>
      </foaf:Person>
    </chump:contributor>
    <chump:contributedAt>2003-06-24 17:03</chump:contributedAt>
    <title>foaf:knows weblogs!</title>
    <description>danbri: Cool :); danbri: Another reason not to forget my
```

```

PGP passphrase. I should re-sign my foaf file now this and foafbot are
taking notice of the signature...; bitsko: FoafCheck is a small Python
script, using Sean Palmer's standalone rdf parser, to retrieve a FOAF
file, verify its signature, and return some interesting properties
from it.; bitsko: FoafCheck is similar to the PHP implementation by
Eric Sigler and Perl XML::FOAF by Ben Trott.; bitsko: I've modified
the Blossom writeback plugin to accept a FOAF URI as the "homepage"
and supply the remaining fields from the FOAF.; bitsko: There is a
question surrounding the use of rdf:seeAlso from the foaf:Person to
connect the given URI to the foaf:Person, followups to the rdfweb-dev
announcement.; danbri: See also danbri's notes on foaf-check.;
mortenf: More notes
http://ilrt.org/discovery/chatlogs/foaf/2003-06-24.html#T13-27-45 on
authentication in #foaf; bitsko: More details on Foaf Identity
Assurance.; (2003-06-24 17:03)
</description>
</item>
<item rdf:about="http://rdfweb.org/">
  <link>http://rdfweb.org/</link>
  <chump:contributor>
    <foaf:Person>
      <foaf:nick>danbri</foaf:nick>
    </foaf:Person>
  </chump:contributor>
  <chump:contributedAt>2003-06-24 16:59</chump:contributedAt>
  <title>RDFWeb/FOAF site</title>
  <description>danbri: Somewhat tidied up, supressing wilful
  obscurity. And experimenting with actually using my Movable Type
  weblog installation.; danbri: Of #rdfig interest, Spring v1.3.1
  for MacOS X just shipped with more (XSLT-based) RDF support --
  loads FOAF files from Web.; danbri: See subsequent rdfweb-dev
  thread on making FOAF/RDF more friendly to XSLT applications.;
  danbri: In that thead, on FOAF/RDF/SW adoption in Japan, " FOAF is
  getting to be the first entry point to RDF/Semantic Web for
  ordinary people. o many people say 'I first time feel partly
  understand RDF' or 'This is my first experience to touch SW, wow!'
  in their blogs or diary pages." ; danbri: "And, people even learn
  XSLT first time in order to render their FOAF/RDF."; (2003-06-24
  16:59)
</description>
</item>
<item rdf:about="http://www.w3.org/2003/03/semantic-tour">
  <link>http://www.w3.org/2003/03/semantic-tour</link>
  <chump:contributor>
    <foaf:Person>
      <foaf:nick>DanC</foaf:nick>
    </foaf:Person>
  </chump:contributor>
  <chump:contributedAt>2003-06-24 16:08</chump:contributedAt>
  <title>W3C Semantic Tour, June 2003</title>
  <description>DanC: presentation materials? notes? trip
  reports?; DanC: some
  http://swordfish.rdfweb.org/photos/2003/06/12/index.htmlphotos of
  the london part from libby; DanC: RDF in Real Life - Some examples
  of RDF applications by Dominique Haza{x{00EB}l-Massieux; DanC:
  Dom's slides are awesome... esp "Benefits of Semantic Web
  Technologies" and "Not tomorrow, today" and "TR automation (2)";
  DanC: ooh... "Decentralized data management"; (2003-06-24 16:08)
</description>
</item>
<item rdf:about="http://www.w3.org/News/2003#item105">
  <link>http://www.w3.org/News/2003#item105</link>
  <chump:contributor>
    <foaf:Person>
      <foaf:nick>danbri</foaf:nick>
    </foaf:Person>
  </chump:contributor>
  <chump:contributedAt>2003-06-24 14:11</chump:contributedAt>
  <title>SOAP Version 1.2 Is a W3C Recommendation</title>
  <description>danbri: Rejoice!; danbri: On my summer
  wishlist: find time to revisit their graph encoding syntax.;
  danbri: See also the primer entry on using other Encoding Schemes

```

```
    for an RDF-within-SOAP example.; (2003-06-24 14:11)
  </description>
</item>
</rdf:RDF>
```

weitere Beispiele:

- [meerkat](#)
- [NewsIsFree](#)

20.4. RDF Schema

- RDF Vocabulary Description Language 1.0: RDF Schema
- Verwendet RDF
- Unterscheidet Klassen und Eigenschaften

Namensräume: `rdf` = `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs` = `http://www.w3.org/2000/01/rdf-schema#`

Klassen (classes)

'Gruppen' von Ressourcen werden als Klassen bezeichnet

- `rdfs:Resource` alle 'Dinge' sind Ressourcen
- `rdfs:Class`, die Ressourcen, die Klassen sind
- `rdfs:Literal`, die Klasse der Literalwerte (Zeichenketten, Zahlen), können Typen haben
- `rdfs:Datatype`, die Klasse der Datentypen
- `rdf:XMLLiteral`, die Klasse der XML Literale
- `rdf:Property`, die Klasse der RDF Properties
-

Eigenschaften (properties)

Properties beschreiben die Beziehung zwischen Subjekt- und Objekt-Ressourcen

- `rdfs:range` Werte einer Property sind Elemente (Exemplare / Instanzen) einer Klasse (Bildbereich einer Beziehung)
- `rdf:type` eine Ressource ist eine Klasse
- `rdfs:subClassOf` (Unter-)Beziehung zwischen Klassen
- `rdfs:subPropertyOf` (Unter-)Beziehung zwischen Properties
- `rdfs:label` von Menschen lesbarer Namen
- `rdfs:comment` von Menschen lesbare Beschreibung
-

sonstiges Vokabular

- `rdfs:Container` Superklasse von `rdf:Bag`, `rdf:Seq`, `rdf:Alt`
- `rdf:Bag` ungeordneter Container
- `rdf:Seq` geordneter Container
- `rdf:Alt` Container von Alternativen
- `rdf:List`

- `rdf:first` erstes Element einer RDF Liste
- `rdf:rest` Rest einer RDF Liste ohne erstes Element
- `rdf:`

Reification Vocabulary

das Vokabular zur Beschreibung und Spezifikation von RDF

- `rdf:Statement` ein RDF Statement
- `rdf:subject` ein RDF Subjekt
- `rdf:predicate` ein RDF Prädikat
- `rdf:object` ein RDF Objekt

Hilfseigenschaften

- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `rdf:value`

20.5. Web Ontology Language (OWL)

- **Ontologie:**
Konzeptionalisierung und Kodierung von Expertenwissen
strukturiertes Glossar / vereinbartes Vokabular
- **DARPA Agent Markup Language (DAML)**
- **Ontology Interference Language (OIL)**
- **DAML + OIL W3C Note Dez. 2001**
- **OWL-Lite:**
Konzepte wie OWL DL, aber Kardinalitäten nur 0 oder 1
- **OWL-DL (Description Language):**
Mächtigkeit wie bei Aussagenlogik (Logik 1. Stufe)
Berechenbarkeit, Entscheidbarkeit
- **OWL-Full:**
Mächtiger wie Aussagenlogik (Logik 2. Stufe)
evtl. Unentscheidbare Aussagen

Namensraum: `owl` = `http://www.w3.org/2002/07/owl#`

OWL Lite

- `owl:Class` definiert eine Klasse
vordefinierte Klassen `owl:Thing` und `owl:Nothing`
- `owl:Individual` definiert ein Exemplar einer Klasse
- aus RDF und RDFS `rdfs:subClassOf`, `rdf:Property`, `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`
- `owl:equivalentClass`, `owl:equivalentProperty` `owl:sameAs` `owl:differentFrom` Gleich- und Ungleichheits-Beziehungen
- `owl:ObjectProperty`, `owl:DatatypeProperty`, `owl:inverseOf`, `owl:TransitiveProperty`, `owl:SymmetricProperty`: Eigenschaften von Properties
- `owl:Restriction`, `owl:cardinality` (nur 0, 1) Einschränkungen von Properties

- `owl:intersectionOf` Durchschnitt von Klassen
- alle Datentypen von XML Schema `xsd`:
- Header Informationen `owl:Ontology`, `owl:imports`
- Annotationen `rdfs:label`, `rdfs:seeAlso`

OWL-DL und OWL-Full

Zusätzlich zu OWL-List werden folgende Dinge definiert.

- `owl:oneOf`
- `owl:equivalentClass` `rdfs:subClassOf` für Ausdrücke, die Klassen enthalten
- `owl:unionOf`, `owl:complementOf`
- `owl:cardinality` beliebige Zahlen

Beispiel [Bier](#) Ontologie.

Zusammenfassung und Ausblick

- RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10. Februar 2004
- Publishing Requirements for Industry Standard Metadata (PRISM)
- Semantic Web
- Web Ontology Language (OWL)
DAML und OIL
- Ableitung von Eigenschaften aus RDF Behauptungen
- Digital Object Identifier (DOI)

10.1000.99/ISBN-3-932588-28-2

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sun Jul 18 16:59:10 CEST 2004

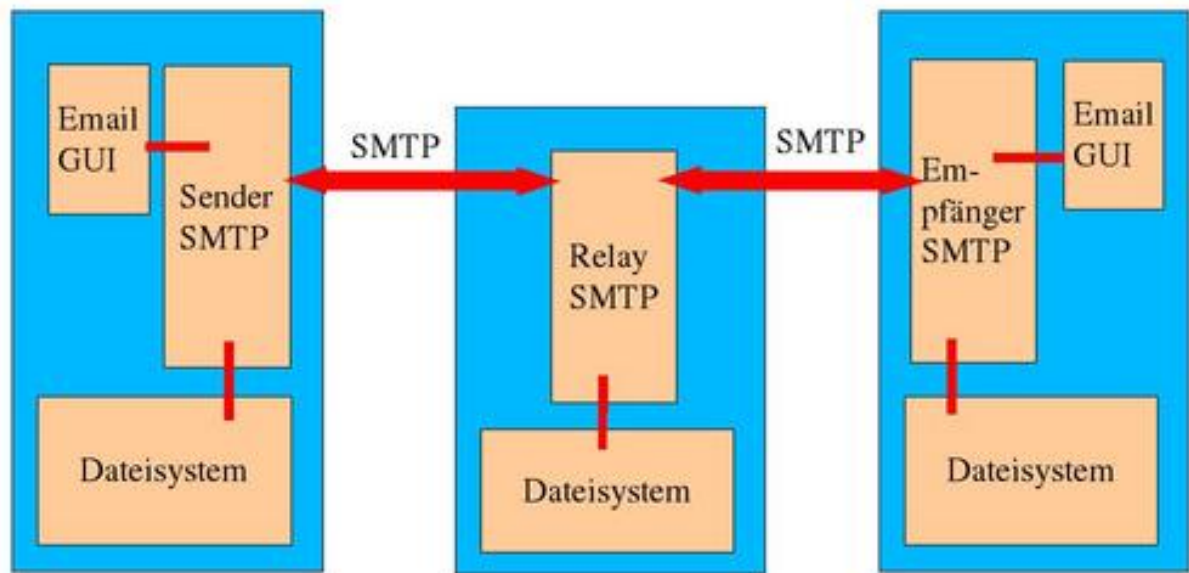
21. Email

- Einleitung
 - Simple Mail Transfer Protocol (SMTP)
 - Nachrichtenformat
 - Multipurpose Internet Mail Extension (MIME)
-

21.1. Einleitung

- Übertragungsprotokoll spezifiziert in RFC 821 vom August 1982
- Nachrichtenformat spezifiziert in RFC 822 vom August 1982
- Übertragung von e-Mail zuverlässig und effizient
- Nutzung von verschiedenen Transport Protokollen:
TCP/IP, NCP, NITS, X-25
- Transport über Zwischenstationen (Relays)
- Store und Forward Konzept
- Nachrichtenformat nur ASCII Text
- andere Formate mit MIME:
Multipurpose Internet Mail Extensions
- OSI Konkurrenzprotokoll: X.400
- Erweitertes Übertragungsprotokoll ESMTP (SMTP Service Extensions) spezifiziert in RFC 1869 vom November 1995

Die Architektur der SMTP Email Software zeigt folgendes Bild.



SMTP Architektur

21.2. Simple Mail Transfer Protocol (SMTP)

- Ablaufbeispiel
- Kommandos
- Antworten

Neuere Versionen des Protokolls sind ESMTP oder SMTP over TLS.

SMTP definiert die heute überall bekannten Email-Adressen:

```
empfaenger@mail.domain.de
```

Beispiel

Der Ablauf einer SMTP Übertragung ist in folgender Tabelle gezeigt.

Client Aktion	Server Aktion
TCP/IP Verbindung zu Port 25	
	220 server.domain.de ESMTP
HELO client.dom.de	
	250 server.domain.de
MAIL FROM: <u@client.dom.de>	
	250 Ok
RCPT TO: <e@mail.domain.de>	

Email

	250 Ok
DATA	
	354 End data with <CR><LF>.<CR><LF>
Text der Email	
.	
	250 Ok
QUIT	
	221 Bye
	Abbau der TCP/IP Verbindung

Beispiel einer SMTP Kommunikation mit der 'rumms'.

```
telnet rumms 25
Trying 134.155.50.52...
Connected to rumms.
Escape character is '^]'.
HELO krabel-wh.isdn.uni-mannheim.de
220 SMTPSERVER ESMTP der UNIVERSITAET MANNHEIM;
    Sun, 11 May 2004 20:16:35 +0200 (MEST)
250 rumms.uni-mannheim.de Hello p3ppp226.rz.uni-mannheim.de
    [134.155.17.226], pleased to meet you
MAIL FROM: krabel@p3ppp226.rz.uni-mannheim.de
250 2.1.0 krabel@p3ppp226.rz.uni-mannheim.de... Sender ok
RCPT TO: krabel@rz.uni-mannheim.de
250 2.1.5 krabel@rz.uni-mannheim.de... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hallo Heinz,
dies ist eine Test mit telnet smtp.
Gruesse, heinz
.
250 2.0.0 h4BIGZs9017920 Message accepted for delivery
QUIT
221 2.0.0 rumms.uni-mannheim.de closing connection
Connection closed by foreign host.
```

Ergebnis der Mailzustellung durch die 'rumms'.

```
From krabel@p3ppp226.rz.uni-mannheim.de Sun May 11 20:18:20 2004
Return-Path: <krabel@p3ppp226.rz.uni-mannheim.de>
Received: from rumms.uni-mannheim.de
    (rumms.uni-mannheim.de [134.155.50.52])
    by krabum2.rz.uni-mannheim.de (8.12.7/8.12.7/SuSE Linux 0.6)
    with ESMTP id h4BIIKMs013655
    for <krabel@krabum2.rz.uni-mannheim.de>;
    Sun, 11 May 2004 20:18:20 +0200
Received: from krabel-wh.isdn.uni-mannheim.de
    (p3ppp226.rz.uni-mannheim.de [134.155.17.226])
    by rumms.uni-mannheim.de (8.12.9/8.12.9) with SMTP id h4BIGZs9017920
    for krabel@rz.uni-mannheim.de; Sun, 11 May 2004 20:17:36 +0200 (MEST)
Message-Id: <200405111817.h4BIGZs9017920@rumms.uni-mannheim.de>
X-Virus-Scanned: by amavisd-new
X-Spamblock-maybe: undisclosed recipients
From: krabel@p3ppp226.rz.uni-mannheim.de
To: undisclosed-recipients:;
Date: Sun, 11 May 2004 20:16:35 +0200 (MEST)

Hallo Heinz,
dies ist eine Test mit telnet smtp.
Gruesse, heinz
```

Kommandos

HELO sender.host

Identifikation des sendenden SMTP Programms

EHLO sender.host

Identifikation des sendenden SMTP Programms und umschalten auf ESMTP Protokoll

MAIL FROM: <sender@host>

Beginn einer Emailübertragung für einen Absender

RCPT TO: <receiver@domain>

Bezeichnung eines Empfängers der Email, bei mehreren Empfängern folgen weitere RCPT Kommandos

DATA

Hiernach folgen die eigentlichen Textdaten der Email bis zur Zeichenfolge <CRLF>.<CRLF>. Kommt diese Zeichenfolge in dem Text vor, wird sie als <CRLF> . <CRLF> gesendet. Der Empfänger muss den zusätzlichen Punkt wieder entfernen. Der Text darf nur 7-bit ASCII Zeichen enthalten.

RSET

Zurücksetzen aller aktuellen Email Informationen von FROM oder TO Kommandos

VERFY zeichenkette

Bitte um Überprüfung, ob die angegebene Zeichenkette zu einem existierenden Benutzer gehört

QUIT

Beenden der Transportverbindung

TURN

wechseln der Rollen zwischen Sender- und Empfänger-SMTP

NOOP

keine Aktion verlangt, der Zustand der Verbindung ändert sich nicht

Kommandos von ESMTP, die von rumms unterstützt werden:

```
EHLO localhost
250-rumms.uni-mannheim.de Hello x.uni-mannheim.de [134.155.x.x],
    pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE 50000000
250-DSN
250-AUTH DIGEST-MD5 CRAM-MD5 PLAIN LOGIN
250-DELIVERBY
250 HELP

HELP
214-2.0.0 This is sendmail version 8.12.11
214-2.0.0 Topics:
214-2.0.0      HELO      EHLO      MAIL      RCPT      DATA
214-2.0.0      RSET      NOOP      QUIT      HELP      VRFY
214-2.0.0      EXPN      VERB      ETRN      DSN      AUTH
214-2.0.0      STARTTLS
214-2.0.0 For more info use "HELP <topic>".
214-2.0.0 To report bugs in the implementation send email to
214-2.0.0      sendmail-bugs@sendmail.org.
214-2.0.0 For local information send email to Postmaster at your site.
214 2.0.0 End of HELP info

HELP AUTH
214-2.0.0 AUTH mechanism [initial-response]
214-2.0.0      Start authentication.
214 2.0.0 End of HELP info

HELP STARTTLS
214-2.0.0 STARTTLS
```

```
214-2.0.0      Start TLS negotiation.  
214 2.0.0 End of HELP info
```

Antworten

Auf jedes Kommando folgt genau eine Antwort. Die Antwort besteht aus einer 3-ziffrigen Statusnummer (für Programme) und einer erklärenden Zeichenkette (für Menschen). Eine Auswahl der Antworten aus RFC 821 ist im folgenden zu sehen.

```
211 System status, or system help reply  
220 <domain> Service ready  
221 <domain> Service closing transmission channel  
250 Requested mail action okay, completed  
251 User not local; will forward to <forward-path>  
  
354 Start mail input; end with <CRLF>.<CRLF>  
  
421 <domain> Service not available,  
      closing transmission channel  
450 Requested mail action not taken: mailbox unavailable  
      [E.g., mailbox busy]  
451 Requested action aborted: local error in processing  
452 Requested action not taken: insufficient system storage  
  
500 Syntax error, command unrecognized  
      [This may include errors such as command line too long]  
501 Syntax error in parameters or arguments  
502 Command not implemented  
503 Bad sequence of commands  
504 Command parameter not implemented  
550 Requested action not taken: mailbox unavailable  
      [E.g., mailbox not found, no access]  
551 User not local; please try <forward-path>  
552 Requested mail action aborted: exceeded storage allocation  
553 Requested action not taken: mailbox name not allowed  
      [E.g., mailbox syntax incorrect]  
554 Transaction failed
```

Die Fehlernummern für die erste und zweite Ziffer sind nach folgendem System klassifiziert.

- **1xx**: positive vorläufige Antwort. Wird nicht verwendet.
- **2xx**: Antwort über den positiven Abschluss einer Aktion.
- **3xx**: positive Zwischenantwort, d.h. bis jetzt ist alles OK aber es fehlen weitere Kommandos.
- **4xx**: negative Zwischenantwort, d.h. bis jetzt ist die Anfrage nicht OK oder kann nicht bearbeitet werden.
- **5xx**: Antwort über den negativen Ausgang einer Aktion oder die Ablehnung des Kommandos.
- **x0x**: bezieht sich auf die Syntax
- **x1x**: zur allgemeinen Information
- **x2x**: bezieht sich auf die (TCP/IP) Verbindung
- **x5x**: bezieht sich auf die SMTP Verarbeitung

21.3. Nachrichtenformat

Der Text der Email besteht aus verschiedenen Header-Teilen, die u.A. den Empfänger und den Absender bezeichnen, und dem eigentlichen ASCII-Text der Nachricht.

To:

Email-Adressen der Empfänger

Cc:

Email-Adressen der Empfänger von Kopien

Bcc:

Email-Adressen der Empfänger von blinden Kopien, d.h. Empfänger über die die To- und Cc-Empfänger nicht informiert werden

From:

verantwortlicher Absender der Email

Sender:

tatsächlicher Absender der Email

Received:

von jeder Zwischenstelle beim Transport wird dies in einem solchen Header vermerkt (ist in SMTP definiert)

Return-Path:

Weg der Email zurück zum Absender (ist in SMTP definiert)

Date:

Datum und Uhrzeit des Versendens der Email

Reply-To:

Email-Adresse, an die die Antworten gesendet werden sollen

Message-Id:

eindeutige Identifikation der Email

In-Reply-To:

Identifikation der Email auf die sich diese Antwort bezieht

Subject:

kurzer Betreff der Email

X-bezeichner:

Header der nicht im RFC 822 definiert ist

X-Virus-Scanned:

Information über einen Virencheck dieser Email

X-Spamblock-maybe:

Information über eventuellen Spam-Inhalt

Beispiel einer Emaildatei mit z.Z. üblichen Headerzeilen.

```
From owner-webmaster@listserv.uni-mannheim.de Sun May 11 14:43:45 2004
Return-Path: <owner-webmaster@listserv.uni-mannheim.de>
Received: from rumms.uni-mannheim.de
    (rumms.uni-mannheim.de [134.155.50.52])
    by krabum2.rz.uni-mannheim.de (8.12.7/8.12.7/SuSE Linux 0.6)
    with ESMTTP id h4BChjMs012633
    for <xxxx@kyyy.rz.uni-mannheim.de>;
    Sun, 11 May 2004 14:43:45 +0200
Received: from warum.uni-mannheim.de
    (warum.uni-mannheim.de [134.155.50.51])
    by rumms.uni-mannheim.de (8.12.9/8.12.9)
    with ESMTTP id h4BChis9000673;
    Sun, 11 May 2004 14:43:44 +0200 (MEST)
Received: (from major@localhost)
    by warum.uni-mannheim.de (8.11.2/8.11.2) id h4BChie22219
    for webmaster-outnew; Sun, 11 May 2004 14:43:44 +0200 (MEST)
X-Authentication-Warning: warum.uni-mannheim.de: major set sender to
    owner-webmaster@listserv.uni-mannheim.de using -f
Received: from rumms.uni-mannheim.de (rumms.uni-mannheim.de [134.155.50.52])
    by warum.uni-mannheim.de (8.11.2/8.11.2) with ESMTTP id h4BChh122214
    for <webmaster@warum.uni-mannheim.de>;
    Sun, 11 May 2004 14:43:43 +0200 (MEST)
Received: from mail.gmx.net (mail.gmx.de [213.165.64.20])
    by rumms.uni-mannheim.de (8.12.9/8.12.9)
    with SMTP id h4BChgs9000627
    for <webmaster@bwl.uni-mannheim.de>;
    Sun, 11 May 2004 14:43:43 +0200 (MEST)
```

```

Received: (qmail 28399 invoked by uid 65534); 11 May 2004 12:43:35 -0000
Received: from pD9ED592C.dip.t-dialin.net (EHLO pacomp) (217.237.89.44)
  by mail.gmx.net (mp016-rz3) with SMTP; 11 May 2004 14:43:35 +0200
Message-ID: <000601c317ba$d005c1e0$0100000a@pacomp>
MIME-Version: 1.0
Content-Type: multipart/alternative;
  boundary="-----_NextPart_000_0003_01C317CB.8AB00FE0"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.50.4133.2400
X-MimeOLE: Produced By Microsoft MimeOLE V5.50.4133.2400
X-Virus-Scanned: by amavisd-new
Precedence: bulk
X-SpamBlock-maybe: Content-Type multipart/html
From: "xxxxxx xxxxx" <marxmax@gmx.ch>
Sender: owner-webmaster@listserv.uni-mannheim.de
To: <webmaster@bwl.uni-mannheim.de>
Subject: Wirtschaftsinformatik nicht gefunden!
Date: Sun, 11 May 2004 14:42:26 +0200

```

21.4. Multipurpose Internet Mail Extension (MIME)

Da SMTP als Emailinhalt nur 7-bit ASCII erlaubt ist einzusätzliches Protokoll notwendig, das den unerlaubten Inhalt geeignet codiert.

MIME ist kompatibel zu SMTP, d.h. MIME Inhalte können normal in einer SMTP Email transportiert werden. MIME wird auch im HTTP Protokoll verwendet.

MIME ist in RFC 1521 spezifiziert. MIME definiert u.A. folgende neue Header.

MIME-Version:

bezeichnet die verwendete Version der MIME Spezifikation

Content-Description:

Menschen lesbare Beschreibung des Inhalts

Content-Id:

eindeutiger Bezeichner für einen Abschnitt im MIME Teil

Content-Transfer-Encoding:

spezifiziert, wie der Inhalt nach 7-bit ASCII transformiert wurde.

- *base64*: dabei werden je 3 Byte auf 4 mal 6-Bit aufgeteilt und diese als 7-bit ASCII verwendet
- *quoted-printable*: dabei werden 7-bit Zeichen normal verwendet und 8-bit Zeichen werden als =hh codiert, wobei 'hh' für die zwei hexadezimal Ziffern des Zeichens steht.

Content-Type: type/subtype

spezifiziert den Datentyp des Inhalts, klassifiziert nach Haupt- und Untertyp. Siehe den Abschnitt über HTTP. Der Typ multipart wird z.B. zum Anfügen von Attachements oder HTML-Inhalt verwendet.

Beispiel einer Emaildatei mit Attachements.

```

MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="-----070909050200090407080406"
...

This is a multi-part message in MIME format.
-----070909050200090407080406
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: quoted-printable

Hallo Herr ...,

```

```
F=FCr Detailfragen stehe ich Ihnen nat=FCrlich jederzeit
gerne pers=F6nlich zur Verf=FCgung.

Viele Gr=FCDFe=20

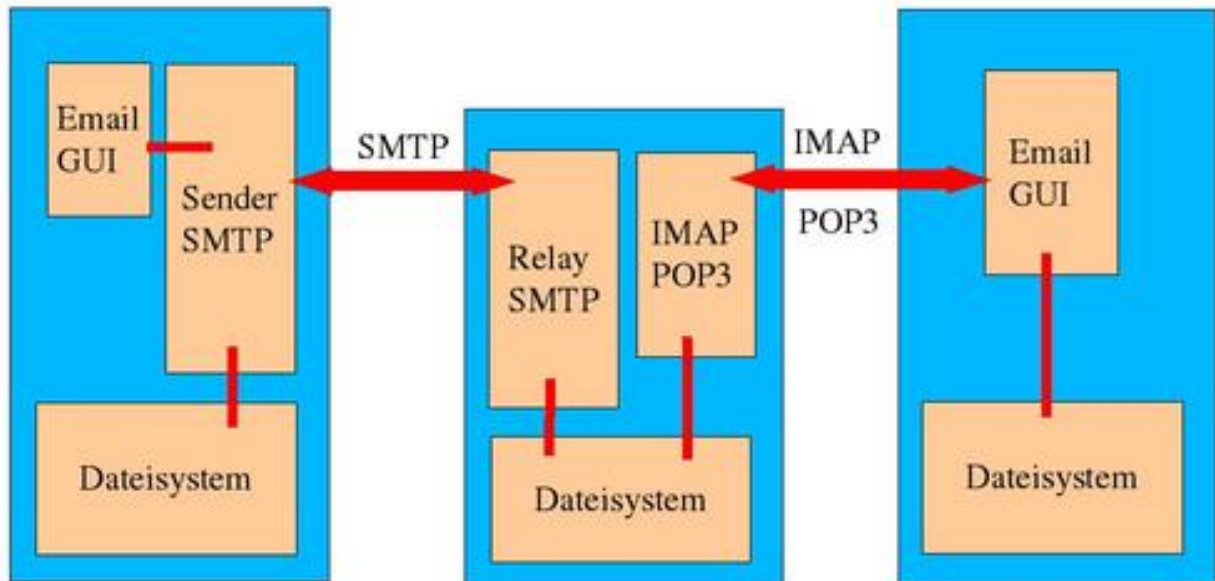
...
-----070909050200090407080406
Content-Type: application/pdf;
  name="xxx_2_neu.pdf"
Content-Transfer-Encoding: base64
Content-Disposition: inline;
  filename="xxx_2_neu.pdf"

JVBERi0xLjMNJeLjz9MNCjMyIDAgb2JqDTw8IA0vTGluZWYyaXplZCAxIA0vTyAzNCANL0gg
WyAyMjcziDQzNCBdIA0vTCA4NTgwMzEgDS9FIDgzNDU0NiANL04gMiANL1QgODU3MjcziA0+
...
XQ0+Pg1zdGFydHhyZWYNTMTczDSU1RU9GDQ==
-----070909050200090407080406--
```

Email Zustellung

- durch Email Reader oder GUI
- Post Office Protocol (POP3), RFC 1225
- Interactive Mail Access Protocol (IMAP), RFC 1064
- Web-Interface, u.U. mit HTTPS
- Weiterleitungen, automatische Antworten

Die Architektur der IMAP und POP3 Email Software zeigt folgendes Bild.



SMTP mit IMAP / POP3 Architektur

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Sun Jun 13 12:27:20 CEST 2004

22. PGP und Kryptographie

- Sicherheitsaspekte
 - Einführung Kryptographie
 - Pretty Good Privacy (PGP)
-

22.1. Sicherheitsaspekte

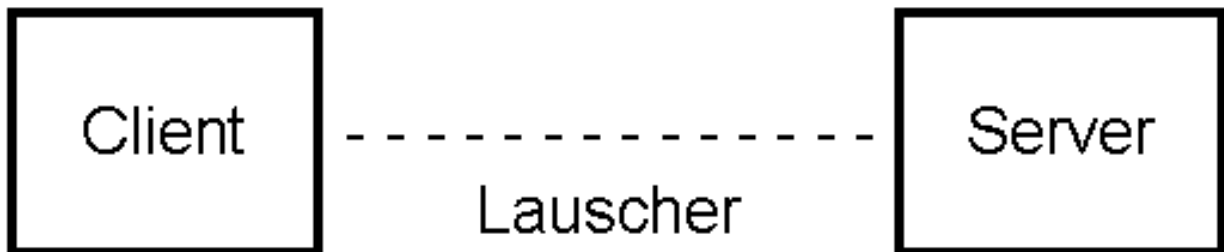
- Internet zwischen fremden Partnern ist unsicher
- Internet Transport Protokoll beinhaltet keine Mechanismen zur Datensicherheit
- Übertragung kann abgehört und verfälscht werden

==> Problem bei vertraulichen Informationen

- Mails
- Passwörter
- Kreditkarteninformationen
- Geldprotokolle

==> sichere Übertragung ist erforderlich um Nutzungsmöglichkeiten des Internets zu erhöhen

Sicherheitsbegriffe



Pakete können

- abgehört
- gefälscht
- verändert
- dupliziert
- gelöscht

werden.

Sicherheit ist

- Vertraulichkeit
- Authentizität
- Integrität

Client/Server selber müssen **sicher** sein

- Firewalls
- physikalisch gesichert
- "Problem Mensch"

22.2. Einführung Kryptographie

Transformation von Nachrichten (messages, plain text) in verschlüsselte Nachrichten (cipher text) mit Hilfe von Algorithmen / Programmen (cipher suites) und Schlüsseln (keys).

- geheime Algorithmen (-> unpraktikabel)
Standard Versuch bei Banken und Telecoms
- bekannte Algorithmen mit geheimen Schlüsseln
im internationalen Security Bereich

Kryptographische Basialgorithmen

Symmetrisch

- gleicher Schlüssel für Ver- und Entschlüsselung
- sehr schnell
- Problem: Anzahl der Schlüssel bei n Partnern
- Problem: sicherer Austausch der Schlüssel
- Beispiele: DES, RC4, Triple-DES, RC2, Idea, Fortezza (PINs, TANs)

Asymmetrisch

- unterschiedliche Schlüssel für Ver- und Entschlüsselung
- öffentliche und private Schlüssel (public und private key)
- relativ langsam
- bei richtiger Wahl der Parameter sehr / beliebig sicher
- Sicherheit basiert auf der Schwierigkeit der Faktorisierung grosser Zahlen, bzw. der Berechnung des diskreten Logarithmus (in endlichen Körpern)
- Beispiele: RSA, DSA, Diffie-Hellman
- Diffie-Hellman (exponential key exchange): zum Aufbau von sicheren Verbindungen aus einem unsicheren Zustand (allerdings ohne Authentifizierung)

Message Digests

- Prüfsummenbildung (Hash Verfahren)
- Berechnung einer Zahl aus einer Nachricht
- Zahl verändert sich, wenn sich die Nachricht ändert
- die Nachricht kann nicht aus der Zahl rekonstruiert werden
- Beispiele: MD5, SHA, SHA1
- Message Authentication Code (MAC)

Kryptographische Basistechniken

Digitale Signatur, Digitale Unterschriften

- basiert auf Digest (Hash) und asymmetrischer Verschlüsselung
- Berechnung und Verschlüsselung der Prüfsumme
- gewährleistet Authentizität

Certification, Zertifizierung

- benötigt unabhängige Verwalter: Certification Authority (CA), sichert die Authentizität von Schlüsseln zu.
- basiert auf asymmetrischer Verschlüsselung, die CA codiert public keys von Teilnehmern mit ihrem private key
- Server verschickt sein Zertifikat an Client bzw. auch umgekehrt: Client verschickt sein Zertifikat an Server
- Zertifikat bestätigt Echtheit des Servers (Echtheit = Zuordnung Server zu public key)
- public key der CA muß dem Client bekannt sein

Blinde Unterschriften

- löst Anonymitätsproblem bei digitalem Geld
- Unterschrift durch Umschlag mit Kohlepapier
- zB. Zertifizierung von eCash unter Wahrung der Anonymität

22.3. Pretty Good Privacy (PGP)

- Entwickelt von Paul Zimmermann für private Emails
- bietet Datenschutz und Authentifizierung
- bietet digitale Unterschriften
- basiert auf RSA, IDEA und MD5
- Verwaltung von öffentlichen Schlüsseln
- kann als normale SMTP Email versendet werden
- Konkurrenzsoftware: Privacy Enhanced Mail (PEM), RFCs 1421 bis 1424
- OpenPGP als RFC 2440
- Gnu PGP (GPG) implementiert OpenPGP, komplettes Open Source Paket

Unterstützte kryptografische Verfahren bei GnuPG:

öffentliche Schlüssel

RSA, RSA-E, RSA-S, ELG-E, DSA, ELG

Verschlüsselung

3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH

Prüfsummen

MD5, SHA1, RIPEMD160

Arbeitsschritte mit GnuPG

1. Erzeugen eines eigenen Schlüsselpaars mit öffentlichem und privatem Teil
2. Versenden des eigenen öffentlichen Schlüssels
3. Entgegennehmen von öffentlichen Schlüsseln anderer Leute
4. Verwalten der (öffentlichen und privaten) Schlüsseln am Schlüsselbund (key ring)
5. Ver- und Entschlüsseln von Dokumenten
6. Signieren und Verifizieren von Dokumenten
7. Editieren von Schlüsseln, d.h. Ändern von Optionen, z.B. Gültigkeitsdauer verlängern
8. Zurückrufen / Ungültigmachen von Schlüsseln
9. Überprüfen der Schlüssel anderer Leute
10. Verwalten der Vertrauensbeziehungen
11. Nutzung von Schlüssel-Servern

Erzeugen eines Schlüsselpaars mit GnuPG

- Zur Benutzung von (Gnu)PGP muss zunächst ein öffentlicher und ein privater Schlüssel erzeugt werden.
- Die PGP Schlüssel müssen nicht wie bei SSL/TLS von einer (unabhängigen) Instanz zertifiziert werden.
- Jede(r) macht sich eine eigene Bewertung von der Vertrauenswürdigkeit der öffentlichen Schlüssel, die verwendet werden.
- öffentliche Schlüssel, die persönlich z.B. über eine Diskette ausgetauscht werden sind vertrauenswürdiger als Schlüssel, die von einer Web-Seite oder von Dritten bezogen werden.
- dadurch wird ein "web of trust" gebildet

Schlüsselerzeugung mit `gpg --gen-key`

```
> gpg --gen-key

gpg (GnuPG) 1.2.2-rc1-SuSE; Copyright (C) 2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen
gpg: /home/krabel/.gnupg: Verzeichnis erzeugt
gpg: Neue Konfigurationsdatei '/home/krabel/.gnupg/gpg.conf' erstellt
gpg: WARNUNG: Die Optionen in '/home/krabel/.gnupg/gpg.conf' sind in diesem Programm1
auf noch nicht aktiv
gpg: Schlüsselbund '/home/krabel/.gnupg/secring.gpg' erstellt
gpg: Schlüsselbund '/home/krabel/.gnupg/pubring.gpg' erstellt
Bitte wählen Sie, welche Art von Schlüssel Sie möchten:
  (1) DSA und ElGamal (voreingestellt)
  (2) DSA (nur signieren/beglaubigen)
  (5) RSA (nur signieren/beglaubigen)
Ihre Auswahl? 1
Das DSA-Schlüsselpaar wird 1024 Bit haben.
Es wird ein neues ELG-E Schlüsselpaar erzeugt.
      kleinste Schlüssellänge ist 768 Bit
      standard Schlüssellänge ist 1024 Bit
      größte sinnvolle Schlüssellänge ist 2048 Bit
Welche Schlüssellänge wünschen Sie? (1024)
Die verlangte Schlüssellänge beträgt 1024 Bit
Bitte wählen Sie, wie lange der Schlüssel gültig bleiben soll.
  0 = Schlüssel verfällt nie
  <n> = Schlüssel verfällt nach n Tagen
  <n>w = Schlüssel verfällt nach n Wochen
```

```

    <n>m = Schlüssel verfällt nach n Monaten
    <n>y = Schlüssel verfällt nach n Jahren
Wie lange bleibt der Schlüssel gültig? (0) 10
Key verfällt am Sam 24 Mai 2004 11:15:26 CEST
Ist dies richtig? (j/n) j

Sie benötigen eine User-ID, um Ihren Schlüssel eindeutig zu machen; das
Programm baut diese User-ID aus Ihrem echten Namen, einem Kommentar und
Ihrer E-Mail-Adresse in dieser Form auf:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Ihr Name ("Vorname Nachname"): Hanno Krabel
E-Mail-Adresse: krabel@uni-mannheim.de
Kommentar: Testen von PGP
Sie haben diese User-ID gewählt:
    "Hanno Krabel (Testen von PGP) <krabel@uni-mannheim.de>"

Ändern: (N)ame, (K)ommentar, (E)-Mail oder (F)ertig/(B)eenden? F
Sie benötigen ein Mantra, um den geheimen Schlüssel zu schützen.
eingabe des mantra
Wir müssen eine ganze Menge Zufallswerte erzeugen. Sie können dies
unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole irgendetwas
tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.
+++++.....+++++
+++++.....+++++>+++++.....
.....+++++
Wir müssen eine ganze Menge Zufallswerte erzeugen. Sie können dies
unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole irgendetwas
tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.
.+++++.....+++++.....+++++.....+++++.....+++++.....+++++.....+++++
+++.....+++++.....+++++.....+++++.....+++++.....+++++.....+++++
.....+++++^^^
gpg: /home/krabel/.gnupg/trustdb.gpg: trust-db erzeugt
Öffentlichen und geheimen Schlüssel erzeugt und signiert.
Schlüssel ist als uneingeschränkt vertrauenswürdig gekennzeichnet.

pub 1024D/ABDFB5BB 2004-05-14 Hanno Krabel (Testen von PGP) <krabel@uni-mannheim.de>
Schl.-Fingerabdruck = D611 446D 2208 E2EB CA7D 2A50 BAF0 ABDF 9016 B5BB
sub 1024g/FECD33F5 2004-05-14 [verfällt: 2004-05-24]

```

Erzeugung eines Widerrufs für den Schlüssel mit gpg --gen-revoke

```

gpg --output revoke-krabel.asc --gen-revoke krabel
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen

sec 1024D/30066B17 2004-05-18 Hanno Krabel (Testen von GnuPG) <krabel@uni-mannheim.de>

Create a revocation certificate for this key? yes
Please select the reason for the revocation:
  0 = Kein Grund angegeben
  1 = Hinweis: Dieser Schlüssel ist nicht mehr sicher
  2 = Schlüssel ist überholt
  3 = Schlüssel wird nicht mehr benutzt
  Q = Cancel
(Probably you want to select 1 here)
Ihre Auswahl? 1
Enter an optional description; end it with an empty line:
> fuer alle faelle
>
Reason for revocation: Hinweis: Dieser Schlüssel ist nicht mehr sicher
fuer alle faelle
Is this okay? yes

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Hanno Krabel (Testen von GnuPG) <krabel@uni-mannheim.de>"
1024-Bit DSA Schlüssel, ID 30066B17, erzeugt 2004-05-18
eingabe des mantra
gpg: GPG-Agent ist in dieser Sitzung nicht vorhanden
ASCII armored output forced.
Revocation certificate created.

```

Please move it to a medium which you can hide away; if Mallory gets access to this certificate he can use it to make your key unusable. It is smart to print this certificate and store it away, just in case your media become unreadable. But have some caution: The print system of your machine might store the data and make it available to others!

Verteilen des eigenen öffentlichen Schlüssels

Speichern des (eigenen) öffentlichen Schlüssels `gpg --export`

```
gpg --armour --export krabel
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.2-rc1-SuSE (GNU/Linux)

mQGIBD5CCYkRBAC0yeXR92PpxwqAeu5u/4jGN+Tv+Mcr121bJmRhc1NR1bonbWkt
...
CQANLwAACgkQuvCQFqvftbt+7gCgk1W5dmBwdm3h2dSeLpo+Mhea4s4AoJ/TETgm
kY2mUPI2L6CIwU7FU/CK
=/dgh
-----END PGP PUBLIC KEY BLOCK-----
```

Importieren von öffentlichen Schlüsseln

Einfügen eines öffentlichen Schlüssels `gpg --import`

```
gpg --import krabel.pub
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen
gpg: Schlüssel D9689977: "Hanno Krabel (Testen von GnuPG) <krabel@uni-mannheim.de>" Nicht geändert
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:                                unverändert: 1
```

Ver- und Entschlüsseln

Verschlüsseln `gpg --encrypt`

```
gpg --armour --output maildoc.sec --encrypt --recipient krabel maildoc.txt
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen
Datei 'maildoc.sec' existiert bereits. Überschreiben (j/N)? j
```

Verschlüsselte Datei:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.2.2-rc1-SuSE (GNU/Linux)

hQE0A5gA7p1IVxqJEAQA3fPS7jWYAWqF4o5fItzqDldXw/+v1WNC/u1Zj414CpKP
...
4viqnRXzi0hoP6eSC0gZX+A0CW02qMq0Q9efZep7xU+0EnG3q570UFD+oiFZvGsG
gJPk6wC
=vNiC
-----END PGP MESSAGE-----
```

Entschlüsseln mit `gpg --decrypt`

```
gpg --output maildoc.decrypt --decrypt maildoc.sec
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
```

```
Benutzer: "Hanno Krabel (Testen von GnuPG) <krabel@uni-mannheim.de>"
1024-Bit ELG-E Schlüssel, ID B2B73599, erzeugt 2004-05-18 (Hauptschlüssel-ID 30021B17)
eingabe des mantra
gpg: GPG-Agent ist in dieser Sitzung nicht vorhanden
gpg: verschlüsselt mit 1024-Bit ELG-E Schlüssel, ID B2B73599, erzeugt 2004-05-18
"Hanno Krabel (Testen von GnuPG) <krabel@uni-mannheim.de>"
```

Signieren von Dateien und Verifizieren

Signieren mit `gpg --sign` oder `gpg --clearsign` oder `gpg --detach-sign`

```
gpg --armour --output maildoc.sign --sign maildoc.txt
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen

Sie benötigen ein Mantra, um den geheimen Schlüssel zu entsperren.
Benutzer: "Helmut Knebel (Testen von GnuPG) <knebel@uni-mannheim.de>"
1024-Bit DSA Schlüssel, ID 30366B17, erzeugt 2004-05-18

gpg: GPG-Agent ist in dieser Sitzung nicht vorhanden
```

Inhalt der Datei mit Signatur:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.2.2-rc1-SuSE (GNU/Linux)

owGbwMvMwCS4yEfXvAFbtjjjmv1J3LmJmTkp+cl6JRuldifdPrpm5ikeJ5ZUKVSV
...
z0F1/Hd+er34byH14Jd8mpyzyp5/f8MwP1hF8WDu86Uimvwn+8Mi10/rWXM4EAA=
=9+OW
-----END PGP MESSAGE-----
```

Verifizieren erfolgt mit `gpg --verify`

```
gpg --verify maildoc.sign
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/de/faq.html für weitere Informationen
gpg: Unterschrift vom Mon 19 Mai 2004 23:04:49 CEST, DSA Schlüssel ID 30366B17
gpg: Korrekte Unterschrift von "Helmut Knebel (Testen von GnuPG) <knebel@rz.uni-mannheim.de>"
```

Anzeigen und Pflegen der Schlüssel

Anzeigen erfolgt mit `gpg --list-keys` oder `gpg --list-public-keys` oder `gpg --list-secret-keys`

```
gpg --list-keys
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/faq.html für weitere Informationen
/home/heinz/.gnupg/pubring.gpg
-----
pub 1024D/578126CA 2004-06-14 Karl Dall (Komiker) <dall@karl.de>
sub 1024g/8FBEEFC6 2004-06-14 [verfällt: 2004-07-14]

gpg --fingerprint
gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/faq.html für weitere Informationen
/home/heinz/.gnupg/pubring.gpg
-----
pub 1024D/578126CA 2004-06-14 Karl Dall (Komiker) <dall@karl.de>
    Schl.-Fingerabdruck = 549C F204 1680 E453 2081 8823 4024 42B1 5781 26CA
sub 1024g/8FBEEFC6 2004-06-14 [verfällt: 2004-07-14]
```

Die Pflege der Schlüssel erfolgt (auch interaktiv) mit `gpg --edit-key`

```
gpg --edit-key 578126CA
gpg (GnuPG) 1.2.2; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: WARNUNG: Sensible Daten könnten auf Platte ausgelagert werden.
gpg: siehe http://www.gnupg.org/faq.html für weitere Informationen
Geheimer Schlüssel ist vorhanden.

pub 1024D/578126CA  erstellt: 2004-06-14 verfällt: 2004-07-14 Vertrauen: u/u
sub 1024g/8FBEEFC6  erstellt: 2004-06-14 verfällt: 2004-07-14
(1). Karl Dall (Komiker) <dall@karl.de>

Befehl> help
quit      Menü verlassen
save      speichern und Menü verlassen

fpr       "Fingerabdruck" anzeigen
list      Schlüssel und User-IDs auflisten
check     Liste der Signaturen
sign      Den Schlüssel signieren
delsig    Signatur entfernen
expire    Ändern des Verfallsdatums
toggle    Umschalten zwischen Anzeige geheimer und öffentlicher Schlüssel
passwd    Die Passphrase ändern
trust     Den "Owner trust" ändern
revsig    Signaturen widerrufen
disable   Schlüssel abschalten
```

Verwalten der Vertrauensbeziehungen

Die gesammelten öffentlichen Schlüssel können nach Vertrauensleveln klassifiziert werden:

unbekannt

es gibt keine überprüften Informationen über den Schlüssel

keine

es ist bekannt, das der Eigentümer *nicht* vertrauenswürdig ist

marginal

dem Eigentümer des Schlüssels wird im Wesentlichen vertraut

voll

dem Eigentümer des Schlüssels wird voll vertraut

Über diese Klassifizierung können dann auch Schlüssel für gültig eingestuft werden, die man nicht selbst überprüft hat.

Danach ist ein Schlüssel gültig, wenn folgende Bedingungen erfüllt sind:

1. der Schlüssel ist von genügend gültigen Schlüsseln signiert, d.h.
 1. von mir persönlich, oder
 2. durch einen, dem ich *voll* vertraue, oder
 3. die mindestens drei, denen ich *marginal* vertraue
2. die Anzahl der Signierschritte zwischen dem Schlüssel und dem eigenen Schlüssel beträgt weniger als *fünf*

Teil 1 erstellt unter Verwendung eines Seminarvortrags von Robert Schulz.

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Mon Jun 14 19:00:30 CEST 2004

23. SSL und TLS

- Kryptographie und TCP/IP
- Secure Socket Layer (SSL)
- Transport Layer Security (TLS)
- Zertifikate nach X.509
- OpenSSL
- Java SSL (JSSE)

23.1. Kryptographie und TCP/IP

Einsatzpunkte im Schichtenmodell

Schicht	Bemerkungen	Kryptographie
5. Verarbeitung Anwendung	Telnet, FTP, SMTP, NNTP, HTTP	PGP, S-HTTP, HTTP over SSL
4. Transport	TCP, UDP	SSL
3. Vermittlung	IP, Internet Protocol	IPv6
2. Sicherung	Adapter-Karten	
1. Bitübertragung	Leitungen, Elektronik	

Verarbeitungs- / Anwendungsebene

- abgestimmt auf Anwendung
- Reimplementierung für jede Anwendung
- Beispiel: PGP, Secure-HTTP (S-HTTP)

Transportschicht

- einmalige Implementierung
- alle Daten werden verschlüsselt
- Beispiel: Secure Socket Layer (SSL)
- HTTP over SSL: `https://host/path/x.html`

Netzwerkschicht / Vermittlungsebene

- einmalige Implementierung
- alle Daten werden verschlüsselt
- Beispiel: IP version 6 (IPv6)

23.2. Secure Socket Layer (SSL) und Transport Layer Security (TLS)

- implementiert die TCP/IP Socket Programmier-Schnittstelle

- Protokoll zur sicheren Client/Server Datenübertragung
- Standardisiert:
 - SSL 2.0 (von Netscape),
 - SSL 3.0 (Internet Draft von Netscape),
 - TLS 1.0 (Internet Standard, von IETF, RFC 2246)

Ziele

- kryptographische Sicherheit
- Interoperabilität
offenes Protokoll
- Erweiterbarkeit
neue Verfahren können eingebunden werden
- Effizienz
Speichermöglichkeit für ausgehandelte Verbindungen

Vorteile

für alle TCP/IP Anwendungen nutzbar
leichter zu benutzen und zu verwalten
effizient

Nachteile

evtl. Probleme mit Firewalls
erweiterbar nur mit Rücksicht auf TCP/IP

SSL in TCP/IP

Anwendung (telnet, ftp, http, ...)

– Verbindungsaufbau	SSL	– Fragmentierung
– Steuerfunktion		– Komprimierung
		– Numerierung
		– Prüfsummen
		– Verschlüsselung

Transportschicht (TCP)

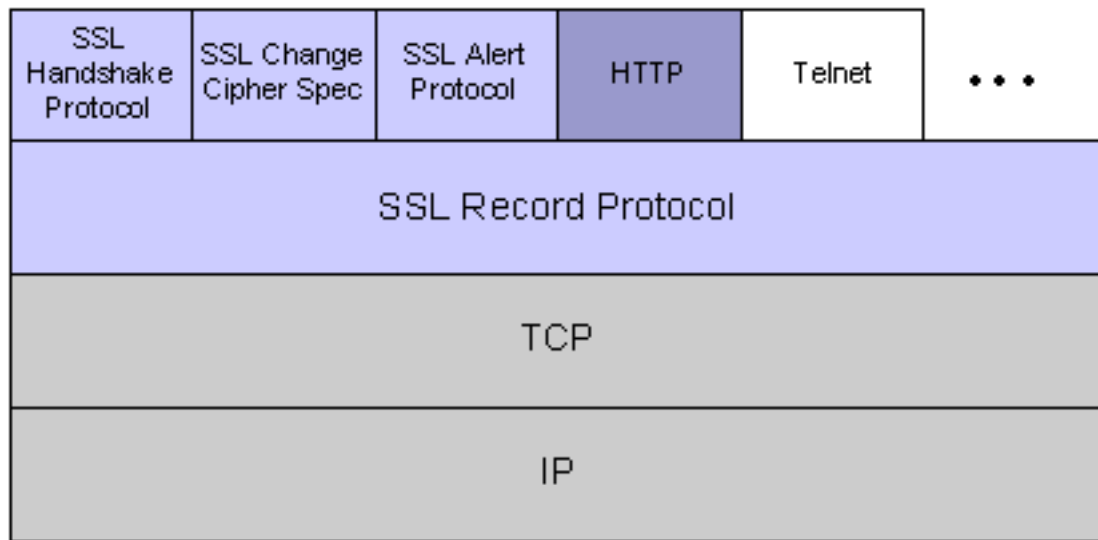
Netzwerkschicht (IP)

Netzwerkzugang

SSL Protokoll liegt zwischen der Transportschicht und der Anwendungsschicht

- aus Sicht der Transportschicht:
SSL ist Anwendung
- aus Sicht der Anwendung:
SSL ist Transportschicht (Socket)

SSL - Aufbau



SSL besteht aus zwei Schichten plus Statusinformationen

Steuerprotokolle

- Verbindungsaushandlung
- schreibt Verbindungsparameter in Status
- austauschbar
- Handshake Protocol, Change Cipher Protocol, Alert Protocol

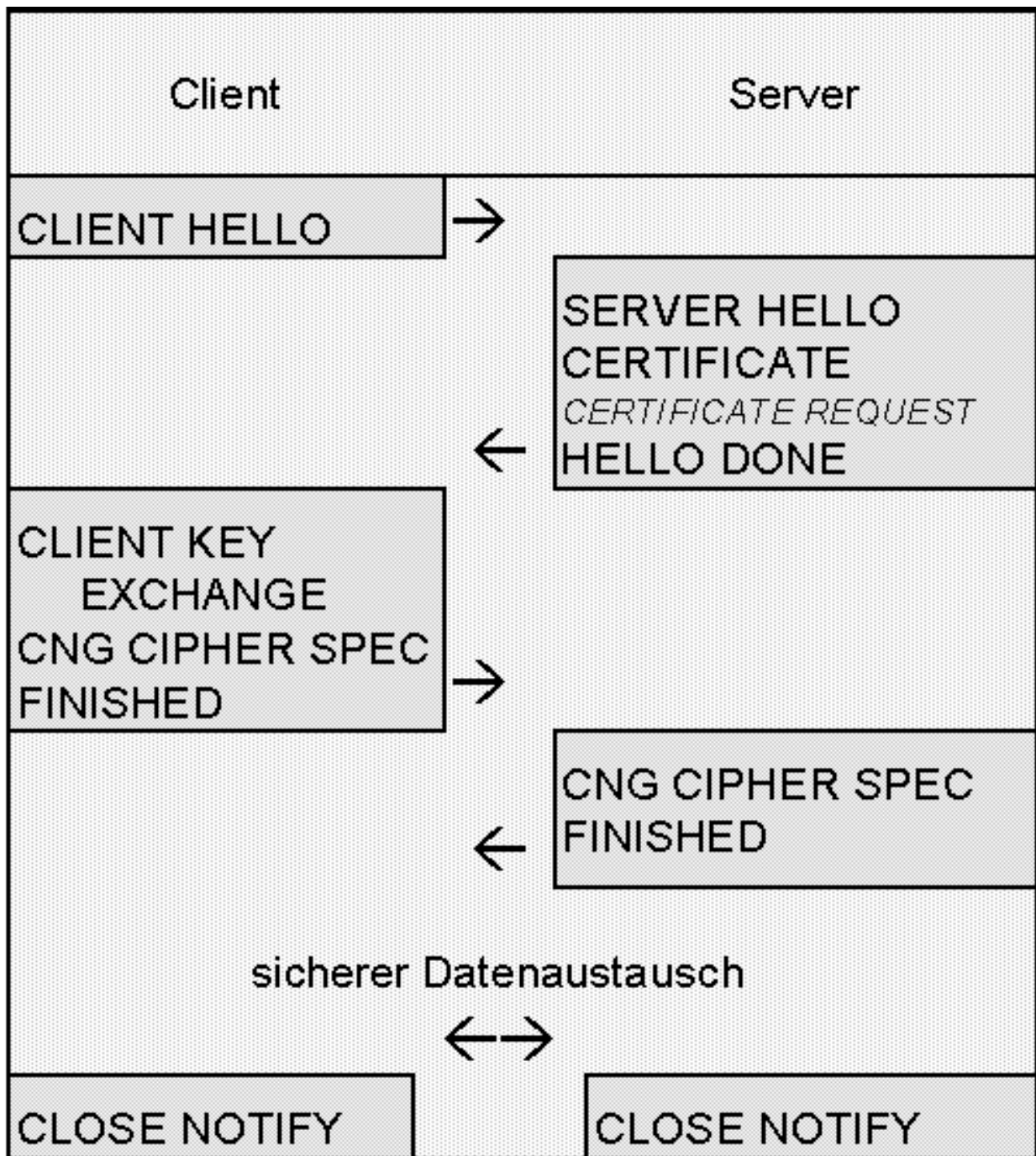
Recordprotokoll

- Kodierung und Transfer entsprechend dem Status

Status

- beinhaltet Verbindungsparameter
- wird mit Nullwerten initialisiert

SSL - Sessions



Ablauf einer Verbindung

SSL - Verbindungsaufbau

durch SSL Handshake Protokoll

- Aushandeln von Verbindungsmodalitäten
Protocol Version, Session Id, Cipher Suite, Compression Method, Random Variables

- Austausch von Zertifikaten (optional)
- Schlüsselaustausch (optional)
- Umstellen der Verbindung entsprechend den ausgehandelten Werten

SSL - Cipher Suites

- Key Exchange Method
wie werden die gemeinsamen symmetrischen Schlüssel ausgetauscht ?
SSL 2.0 nur RSA, SSL 3.0 auch andere
- Cipher for Data Transfer, mit symmetrischen Schlüssel
Stream Ciphers: RC4
Block Ciphers: RC2, DES, Triple-DES, IDEA
- Message Digest Method
MD5, SHA-1, für Message Authentication (MAC)

SSL - Fehlerbehandlung

SSL Alert Protokoll behandelt folgende Fehlerarten

- unerwartete Nachricht
- falsche Prüfsumme
- Entpackungsfehler
- Handshake Fehler
- Kein Zertifikat vorhanden
- Ungültiges Zertifikat
- Nicht unterstütztes Zertifikat
- Ungültiger Parameter

=> Fehlerfall: Nachricht und Verbindungsabbruch

- Fehlernachricht wird verschlüsselt gesandt
- alle Informationen über die Verbindung werden gelöscht

SSL - Implementierungen

- Browser:
Netscape 1.x, 2.x, 3.x, 4.x, 6.x
Microsoft IE 3.x, 4.x, 5.x
Lynx 2.8 mit OpenSSL
- Server:
Netscape, Microsoft
Apache mit mod_ssl und OpenSSL

Probleme:

- Import / Export Restriktionen einzelner Länder
- Beschränkungen in der Schlüssellänge
- Patentierung einiger Verfahren

23.3. Zertifikate nach X.509

Zertifikat (certificate) beglaubigt die Verbindung von einem öffentlichen Schlüssel und der Identifikation einer Person.

Die Personen-Identifikation erfolgt mit einem *Distinguished Name (DN)*.

Die Spezifikation eines DN ist im X.509 Standard festgelegt.

DN Feld	Abkürzung	Bedeutung
Common Name	CN	Name der Person
Organization	O	Firma oder Organisation
Organizational Unit	OU	Abteilung oder Firmenteil
Locality	L	Stadt, Sitz der Organisation
State	ST	Staat, Provinz, Gegend
Country	C	ISO Ländercode

Beispiel:

```
CN=Karl Dall
O=Schlangenöl AG
OU=Brillenabteilung
L=Schlangenbad
ST=Hessen
C=de
```

Ein Zertifikat kombiniert die Informationen über das Individuum mit weiteren Informationen über die Ausgabestelle und diversen Verwaltungsinformationen:

Subject: Individuum

Distinguished Name, Public Key

Issuer: Ausgabestelle

Distinguished Name, Signature

Period of Validity: Geltungsdauer

Not Before Date, Not After Date

Administrative Infos:

Certificate Version, Serial Number

Extended Infos: Zusatzinformationen

Basic Constraints, Netscape Flags, etc.

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
  Signature Algorithm: md5WithRSAEncryption
  Issuer: C=XY, ST=Snake Desert, L=Snake Town, O=Snake Oil, Ltd,
         OU=Certificate Authority, CN=Snake Oil CA/Email=ca@snakeoil.dom
  Validity
    Not Before: Oct 29 17:39:10 2000 GMT
    Not After : Oct 29 17:39:10 2001 GMT
  Subject: C=DE, ST=Germany, L=Weinheim, O=Home, OU=Web Lab,
         CN=krimmel-wh.isdn.uni-mannheim.de/Email=krimmel@rz.uni-mannheim.de
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
    Modulus (1024 bit):
      00:c4:40:4c:6e:14:1b:61:36:84:24:b2:61:c0:b5:
```

```
...
f0:b4:95:f5:f9:34:9f:f8:43
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Alternative Name:
email:krime1@rz.uni-mannheim.de
Netscape Comment:
mod_ssl generated test server certificate
Netscape Cert Type:
SSL Server
Signature Algorithm: md5WithRSAEncryption
12:ed:f7:b3:5e:a0:93:3f:a0:1d:60:cb:47:19:7d:15:59:9b:
...
```

Zertifikate können bei verschiedenen Organisationen (gegen Gebühr) bestellt werden. Zum Beispiel bei Verisign, Thawte, CertiSign oder IKS GmbH.

23.4. OpenSSL

OpenSSL bietet eine offene Implementierung von SSL und TLS.

Arbeitsschritte mit SSL/TLS

Die gewünschten Schlüssel kann man sich mit Hilfe der OpenSSL Kommandozeilen Tools erzeugen.

1. Erzeugen eines eigenen Schlüsselpaars mit öffentlichem und privatem Teil
2. Erzeugen einer Anforderung zur Zertifizierung
3. Zertifizieren von Schlüsseln
4. Testen der verschlüsselten Kommunikation mit dem OpenSSL Server oder Client
5. Einsatz der Zertifikate in Servern (z.B. Web-Server mit HTTPS)
6. Einsatz der Zertifikate in Browsern (auch über HTTPS)

Beispiel Laden eines DFN Zertifikates in den Browser von der [RUM CA](#).

Erzeugen eines Schlüsselpaares

Schlüsselpaare werden mit dem Kommando `openssl genrsa` erzeugt.

```
openssl genrsa -des3 -out user.key 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

Ergebnis:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-EDE3-CBC, DCDD31A379A9E2D6

TlFTGb0fniqLvHIWdmp+faRJA3wYI2WUoiqhGZl85HuMidEK08UNRePwtqRnKNWv
DEUDBiGWLap7bgg9yaWuK8jSruSY0JZGkPQd09t4bD6TQiTXTRkDyc8iIS8R2tbD
...
6DfMmbpuhsaIbYyv3BQfmNDowSeebTLbtU67jvzq6rzGxcvmrCABNIES4B7PMqrY
v0WcUNBSF5+PAMw5V2KsKTKlesuh/Yczh7bkBMAQdJo=
```

```
-----END RSA PRIVATE KEY-----
```

Ergebnis als Text:

```
openssl rsa -noout -text -in myca.key
read RSA key
Enter PEM pass phrase:

Private-Key: (1024 bit)
modulus:
  00:9d:a0:a1:0e:97:55:aa:74:88:3b:05:c0:18:72:
  ...
  58:72:a4:3a:41:46:22:5a:d5
publicExponent: 65537 (0x10001)
privateExponent:
  ...
prime1:
  00:cb:ba:31:10:49:d7:4e:51:99:58:c1:cc:05:54:
  ...
prime2:
  00:c6:12:5f:85:c2:f1:a8:e7:20:00:e4:af:8f:4d:
  ...
exponent1:
  2a:db:a9:94:ae:a4:0f:c2:d4:ca:ba:42:4c:60:cb:
  ...
exponent2:
  02:d3:6a:47:77:43:8
  ...
coefficient:
  5b:58:6d:ef:be:37:45:e7:93:36:fd:ae:a9:86:d6:
  ...
```

Erzeugen einer Anforderung einer Zertifizierung

Zertifikatsanforderungen werden mit dem Kommando `openssl req` erzeugt.

```
openssl req -new -key user.key -out user.csr
Using configuration from /etc/ssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:ba-wue
Locality Name (eg, city) []:mannheim
Organization Name (eg, company) [Internet Widgits Pty Ltd]:uni
Organizational Unit Name (eg, section) []:it
Common Name (eg, YOUR name) []:Karl Dall
Email Address []:kd@al.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:.
An optional company name []:.
```

Ergebnis:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBuTCCASICAQAwETELMAkGA1UEBhMCZGUxDzANBgNVBAgTBmJhLXd1ZTERMA8G
...
skXL47VitW2udc/Mgg==
-----END CERTIFICATE REQUEST-----
```

Ergebnis als Text:

```
openssl req -noout -text -in user.csr
Using configuration from /etc/ssl/openssl.cnf

Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: C=de, ST=ba-wue, L=mannheim, O=uni, OU=it, CN=Karl Dall/Email=kd@al.de
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:9d:a0:a1:0e:97:55:aa:74:88:3b:05:c0:18:72:
          ...
          58:72:a4:3a:41:46:22:5a:d5
        Exponent: 65537 (0x10001)
    Attributes:
      a0:00
  Signature Algorithm: md5WithRSAEncryption
  37:1c:98:34:0f:42:67:84:e9:2a:8c:f3:3c:38:8e:3a:9d:75:
  ...
  f0:ce:20:2b:62:dd:5a:b2:45:cb:e3:b5:62:b5:6d:ae:75:cf:
  cc:82
```

Erzeugen eines selbstsignierten Zertifikats

Zertifikate für den eigenen Schlüssel können selbst erzeugt werden mit dem Kommando `openssl req -x509`.

```
openssl req -new -x509 -key myca.key -out myca.crt
Using configuration from /etc/ssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:de
State or Province Name (full name) [Some-State]:ba-wue
Locality Name (eg, city) []:mannheim
Organization Name (eg, company) [Internet Widgits Pty Ltd]:uni
Organizational Unit Name (eg, section) []:it
Common Name (eg, YOUR name) []:Carl Cert
Email Address []:cert@home.de
```

Ergebnis:

```
openssl x509 -noout -text -in myca.crt
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 0 (0x0)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=de, ST=ba-wue, L=mannheim, O=uni, OU=it, CN=Carl Cert/Email=cert@home.de
    Validity
      Not Before: May 22 20:12:08 2004 GMT
      Not After : Jun 21 20:12:08 2004 GMT
    Subject: C=de, ST=ba-wue, L=mannheim, O=uni, OU=it, CN=Carl Cert/Email=cert@home.de
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:b5:b1:39:fb:4a:ca:1b:af:05:4e:87:20:dd:86:
          ...
          37:ac:91:e3:36:f0:29:f0:21
```

```
      Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Subject Key Identifier:
    C1:F0:F0:91:DC:CD:7E:7A:29:1C:44:0A:F2:64:0F:7E:C9:1F:BE:BE
  X509v3 Authority Key Identifier:
    keyid:C1:F0:F0:91:DC:CD:7E:7A:29:1C:44:0A:F2:64:0F:7E:C9:1F:BE:BE
    DirName:/C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
    serial:00
  X509v3 Basic Constraints:
    CA:TRUE
Signature Algorithm: md5WithRSAEncryption
18:15:56:cb:51:b1:e2:6d:8e:c5:a4:e0:45:4c:57:60:51:6b:
...
ff:8d
```

Testen von SSL-Verbindungen

Die Kommunikation über SSL kann mit den Kommandos `openssl s_server` für einen SSL-Server und `openssl s_client` für einen SSL-Client getestet werden. Zum Beispiel kann mit dem Client eine Verbindung zu einem HTTPS Web-Server hergestellt werden, oder mit dem Server kann ein HTTPS Web-Browser getestet werden.

Starten eines SSL-Servers:

```
openssl s_server -cert myca.crt -key myca.key
Using default temp DH parameters
Enter PEM pass phrase:
ACCEPT
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAGMBBATAfGqGwG2fR8+cuE+pH/IzRoGu7Bao9hDtGuhdXxQwboj1WNoE
MHSAmy7Cin9t6lchz+P0i0XQOpI70+1UUuL8fcPOVFU0NUe3rz80sacXxD1wTrgi
2aEGAgQ+0o1HogQCAGEspAYEBAEAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DES-CBC3-SHA:
DHE-DSS-RC4-SHA:RC4-SHA:RC4-MD5:EXP1024-DHE-DSS-RC4-SHA:EXP1024-RC4-SHA:
EXP1024-DHE-DSS-DES-CBC-SHA:EXP1024-DES-CBC-SHA:EXP1024-RC2-CBC-MD5:
EXP1024-RC4-MD5:EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:DES-CBC-SHA:
EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-CBC-SHA:EXP-DES-CBC-SHA:
EXP-RC2-CBC-MD5:EXP-RC4-MD5
CIPHER is EDH-RSA-DES-CBC3-SHA
Hallo
Wie gehts?
DONE
shutting down SSL
CONNECTION CLOSED
```

Starten eines SSL-Clients:

```
openssl s_client -connect localhost:4433 -state
CONNECTED(00000003)
SSL_connect:before/connect initialization
SSL_connect:SSLv2/v3 write client hello A
SSL_connect:SSLv3 read server hello A
depth=0 /C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
verify error:num=18:self signed certificate
verify return:1
depth=0 /C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
verify return:1
SSL_connect:SSLv3 read server certificate A
SSL_connect:SSLv3 read server key exchange A
SSL_connect:SSLv3 read server done A
SSL_connect:SSLv3 write client key exchange A
SSL_connect:SSLv3 write change cipher spec A
SSL_connect:SSLv3 write finished A
SSL_connect:SSLv3 flush data
SSL_connect:SSLv3 read finished A
---
Certificate chain
0 s:/C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
```

```

i:/C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDTDCCArWgAwIBAgIBADANBgkqhkiG9w0BAQQFADB9MQswCQYDVQQGEWJkZTEP
...
agFM3T0F5bRis9fyA2JspiJTV2RwgzWU4PD1tw6L/40=
-----END CERTIFICATE-----
subject=/C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
issuer=/C=de/ST=ba-wue/L=mannheim/O=uni/OU=it/CN=Carl Cert/Email=cert@home.de
---
No client certificate CA names sent
---
SSL handshake has read 1276 bytes and written 250 bytes
---
New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol    : TLSv1
    Cipher      : EDH-RSA-DES-CBC3-SHA
    Session-ID: 586D9F47CF9CB84FA91FF2334681AEEC16A8F610ED1AE85D5F14306E88E558DA
    Session-ID-ctx:
    Master-Key: 74809B2EC2D67F6DEA5721CFE3CE88E5D03A923BD3E95452E2FC7DC3CE5455343547B7AF3F0EB1A717C43D704E
    Key-Arg     : None
    Start Time: 1053985095
    Timeout     : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
Hallo
Wie gehts?
DONE
SSL3 alert write:warning:close notify

```

23.5. Java Secure Socket Extension (JSSE)

- JSSE ist im JDK 1.4 enthalten.
- JSSE basiert auf der Java Cryptography Architecture (JCA).
- JSSE unterstützt SSL 3.0 und TLS 1.0.
- Die wichtigsten Klassen sind SSLSocket und SSLServerSocket, die wie Socket und ServerSocket verwendet werden können.
- JSSE ist u.A. in folgenden Paketen enthalten: javax.net, javax.net.ssl und javax.security.certs

Beispiel HelloWorld

HelloWorldSSLClient:

```

import java.io.IOException;
import java.net.Socket;
import javax.net.SocketFactory;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.SSLSocket;

public class HelloWorldSSLClient {

    static int port = 40000;
    static String host = "localhost";
    static String msg = "Hallo Welt";

    public static void main(String[] args) {

        if (args.length > 0) host = args[0];
        if (args.length > 1) msg = args[1];
    }
}

```

```

Integer x = new Integer(4711);
SocketFactory sf = SSLSocketFactory.getDefault();

try {
    Socket s = sf.createSocket( host, port );
    System.out.println("Client connected to Server");
    SocketChannel c = new SocketChannel(s);
    c.send( msg );
    c.send( x );
    c.close();
} catch (IOException e) {
    System.out.println("IOException "+e);
}
}

```

HelloWorldSSLServ:

```

import java.io.IOException;
import java.net.Socket;
import java.net.ServerSocket;
import javax.net.ServerSocketFactory;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLServerSocket;

public class HelloWorldSSLServ {

    static int port = 40000;

    public static void main(String[] args) {

        SSLServerSocketFactory ssf =
            (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
        String[] dcs = ssf.getDefaultCipherSuites();
        //String[] dcs = ssf.getSupportedCipherSuites();
        System.out.print("\nciphers = ");
        for (int i = 0; i < dcs.length; i++ ) {
            System.out.print(dcs[i]);
            if ( i < dcs.length-1 ) {
                System.out.print(", ");
            }
        }
        System.out.println("\n");
        try {
            ServerSocket so = ssf.createServerSocket( port );
            System.out.println("Server startet on port "+port);
            while (true) {
                System.out.println("waiting for connection ... ");
                Socket s = so.accept();
                System.out.println("new connection from "
                    + s.getInetAddress() );
                SocketChannel c = new SocketChannel(s);
                System.out.println("Object Stream created ");
                Object o = c.receive();
                System.out.println("message 1 " + o);
                o = c.receive();
                System.out.println("message 2 " + o);
                c.close();
            }
        } catch (IOException e) {
            System.out.println("IOException "+e);
        }
        catch (ClassNotFoundException e) {
            System.out.println("ClassNotFoundException "+e);
        }
    }
}

```

Starten des Servers:

```
java -Djavax.net.ssl.keyStore=inetserver.keys
-Djavax.net.ssl.keyStorePassword=internet
HelloWorldSSLServ

ciphers = SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA,
SSL_DHE_DSS_WITH_DES_CBC_SHA, SSL_RSA_EXPORT_WITH_RC4_40_MD5,
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA,
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA.

Server startet on port 40000
waiting for connection ...
new connection from /127.0.0.1
Object Stream created
message 1 Hallo Welt
message 2 4711
waiting for connection ...
new connection from /127.0.0.1
Object Stream created
message 1 Hallo Welt
message 2 4711
```

Starten des Clients:

```
java -Djavax.net.ssl.trustStore=client.trusts
-Djavax.net.ssl.trustStorePassword=internet
HelloWorldSSLClient
Client connected to Server
```

Verwalten der Schlüssel mit keytool

keytool Optionen:

```
-certreq [-v] [-alias <Alias>] [-sigalg <Sigalg>]
[-file <csr_Datei>] [-keypass <Keypass>]
[-keystore <Keystore>] [-storepass <Storepass>]
[-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-delete [-v] -alias <Alias>
[-keystore <Keystore>] [-storepass <Storepass>]
[-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-export [-v] [-rfc] [-alias <Alias>] [-file <Zert_Datei>]
[-keystore <Keystore>] [-storepass <Storepass>]
[-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-genkey [-v] [-alias <Alias>] [-keyalg <Schlüsselalg>]
[-keysize <Schlüsselgröße>] [-sigalg <Sigalg>]
[-dname <dname>] [-validity <GültigTage>]
[-keypass <Schlüsselpass>] [-keystore <Keystore>]
[-storepass <Storepass>] [-storetype <Storetyp>]
[-provider <Provider_Klassenname>] ...

-help

-identitydb [-v] [-file <idb_Datei>] [-keystore <Keystore>]
[-storepass <Storepass>] [-storetype <Storetyp>]
[-provider <Provider_Klassenname>] ...

-import [-v] [-noprompt] [-trustcacerts] [-alias <Alias>]
[-file <Zert_Datei>] [-keypass <Schlüsselpass>]
[-keystore <Keystore>] [-storepass <Storepass>]
[-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...
```

```
-keyclone [-v] [-alias <Alias>] -dest <Ziel_Alias>
          [-keypass <Schlüsselpass>] [-new <neu_Schlüsselpass>]
          [-keystore <Keystore>] [-storepass <Storepass>]
          [-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-keypasswd [-v] [-alias <Alias>]
           [-keypass <alt_Schlüsselpass>] [-new <neu_Schlüsselpass>]
           [-keystore <Keystore>] [-storepass <Storepass>]
           [-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-list [-v | -rfc] [-alias <Alias>]
      [-keystore <Keystore>] [-storepass <Storepass>]
      [-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...

-printcert [-v] [-file <Zert_Datei>]

-selfcert [-v] [-alias <Alias>] [-sigalg <Sigalg>]
          [-dname <dname>] [-validity <GültigTage>]
          [-keypass <Schlüsselpass>] [-keystore <Keystore>]
          [-storepass <Storepass>] [-storetype <Storetyp>]
          [-provider <Provider_Klassenname>] ...

-storepasswd [-v] [-new <neu_Storepass>]
             [-keystore <Keystore>] [-storepass <Storepass>]
             [-storetype <Storetyp>] [-provider <Provider_Klassenname>] ...
```

Erzeugen eines Keystores `inetserv.keys` und eines Server-Keys:

```
keytool -genkey -alias inetserv -keyalg RSA -validity 60
        -keystore inetserv.keys

Geben Sie das Keystore-Passwort ein: internet
Wie lautet Ihr Vor- und Nachname?
[Unknown]: Karl Dall
Wie lautet der Name Ihrer organisatorischen Einheit?
[Unknown]: Internet-Technologien
Wie lautet der Name Ihrer Organisation?
[Unknown]: Universitaet Mannheim
Wie lautet der Name Ihrer Stadt oder Gemeinde?
[Unknown]: Mannheim
Wie lautet der Name Ihres Bundeslandes oder Ihrer Provinz?
[Unknown]:
Wie lautet der Landescode (zwei Buchstaben) für diese Einheit?
[Unknown]: de
Ist CN=Karl Dall, OU=Internet-Technologien,
    O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
richtig?
[Nein]: ja

Geben Sie das Passwort für <inetserv> ein.
(EINGABETASTE, wenn Passwort dasselbe wie für Keystore):
```

Anzeigen des Inhalts des Keystores:

```
keytool -list -v -keystore inetserv.keys
Geben Sie das Keystore-Passwort ein: internet

Keystore-Typ: jks
Keystore-Provider: SUN

Ihr Keystore enthält 1 Eintrag/-äge.

Aliasname: inetserv
Erstellungsdatum: 06.06.2004
Eintragstyp: keyEntry
Zertifikatskettenlänge: 1
Zertifikat[1]:
Eigentümer: CN=Karl Dall, OU=Internet-Technologien,
    O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Aussteller: CN=Karl Dall, OU=Internet-Technologien,
```

SSL und TLS

```
O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Seriennummer 40c336ed
Gültig ab: Sun Jun 06 17:23:25 CEST 2004
      bis: Thu Aug 05 17:23:25 CEST 2004
Zertifikatfingerabdrücke:
MD5: 2C:E2:49:FE:30:C1:D6:21:D7:F3:C3:97:48:46:32:C7
SHA1: CB:2E:96:F1:2E:A6:00:BB:F2:37:15:27:79:ED:96:8F:88:A6:A2:56
```

Exportieren des Zertifikats für den Server:

```
keytool -export -alias inetserver -keystore inetserver.keys
      -rfc -file server.certs
```

Geben Sie das Keystore-Passwort ein: internet
Zertifikat in Datei <server.certs> gespeichert.

Anzeigen des Zertifikats als Textdatei:

```
cat server.certs
-----BEGIN CERTIFICATE-----
MIICgDCCAekCBEDDNU0wDQYJKoZIhvcNAQEEBQAwgYYxCzAJBgNVBAYTAmRlMRAwDgYDVQQIEwdV
bmtub3duMREwDwYDVQQHEWhNYW5uaGVpbTEeMBwGA1UEChMVVW5pdmVyc2l0YWV0IE1hbm5oZWlt
...
eZqjMBRqfVa0P45SmbqBNk=
-----END CERTIFICATE-----
```

Anzeigen des Zertifikats mit OpenSSL:

```
openssl x509 -noout -text -in server.certs

Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 1086535405 (0x40c336ed)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=de, ST=Unknown, L=Mannheim, O=Universitaet Mannheim,
      OU=Internet-Technologien, CN=Karl Dall
    Validity
      Not Before: Jun  6 15:23:25 2004 GMT
      Not After : Aug  5 15:23:25 2004 GMT
    Subject: C=de, ST=Unknown, L=Mannheim, O=Universitaet Mannheim,
      OU=Internet-Technologien, CN=Karl Dall
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
      Modulus (1024 bit):
        00:e9:74:a8:5a:27:9e:03:6f:bc:12:18:2d:8c:2a:
        ...
        b1:b8:18:9a:f0:0f:d5:10:37
      Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
      58:8d:c8:6e:2b:3c:44:6b:45:e6:c3:65:41:7e:6e:0f:b0:5f:
      ...
      04:d9
```

Anzeigen des Zertifikats mit Java:

```
keytool -printcert -file server.certs

Eigentümer: CN=Karl Dall, OU=Internet-Technologien,
  O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Aussteller: CN=Karl Dall, OU=Internet-Technologien,
  O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Seriennummer 40c336ed
Gültig ab: Sun Jun 06 17:23:25 CEST 2004
      bis: Thu Aug 05 17:23:25 CEST 2004
Zertifikatfingerabdrücke:
```

```
MD5: 2C:E2:49:FE:30:C1:D6:21:D7:F3:C3:97:48:46:32:C7
SHA1: CB:2E:96:F1:2E:A6:00:BB:F2:37:15:27:79:ED:96:8F:88:A6:A2:56
```

Erzeugen eines Truststores client.trusts für den Client und importieren des Server Zertifikats:

```
keytool -import -alias inetserv -file server.certs
-keystore client.trusts

Geben Sie das Keystore-Passwort ein: internet
Eigentümer: CN=Karl Dall, OU=Internet-Technologien,
O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Aussteller: CN=Karl Dall, OU=Internet-Technologien,
O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Seriennummer 40c336ed
Gültig ab: Sun Jun 06 17:23:25 CEST 2004
bis: Thu Aug 05 17:23:25 CEST 2004
Zertifikatfingerabdrücke:
MD5: 2C:E2:49:FE:30:C1:D6:21:D7:F3:C3:97:48:46:32:C7
SHA1: CB:2E:96:F1:2E:A6:00:BB:F2:37:15:27:79:ED:96:8F:88:A6:A2:56
Diesem Zertifikat vertrauen? [Nein]: ja
Zertifikat wurde zu Keystore hinzugefügt.
```

Anzeigen des Inhalts des Truststores:

```
keytool -list -v -keystore client.trusts

Geben Sie das Keystore-Passwort ein: internet

Keystore-Typ: jks
Keystore-Provider: SUN

Ihr Keystore enthält 1 Eintrag/-äge.

Aliasname: inetserv
Erstellungsdatum: 06.06.2004
Eintragstyp: trustedCertEntry

Eigentümer: CN=Karl Dall, OU=Internet-Technologien,
O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Aussteller: CN=Karl Dall, OU=Internet-Technologien,
O=Universitaet Mannheim, L=Mannheim, ST=Unknown, C=de
Seriennummer 40c336ed
Gültig ab: Sun Jun 06 17:23:25 CEST 2004
bis: Thu Aug 05 17:23:25 CEST 2004
Zertifikatfingerabdrücke:
MD5: 2C:E2:49:FE:30:C1:D6:21:D7:F3:C3:97:48:46:32:C7
SHA1: CB:2E:96:F1:2E:A6:00:BB:F2:37:15:27:79:ED:96:8F:88:A6:A2:56
```

Ausblick

- HTTP over SSL
- SMTP mit STARTTLS
- SMTP mit SASL AUTH
- POP mit STLS
- IMAP mit STARTTLS
- Kerberos und AFS
- MIME-SEC
- Simple Authentication and Security Layer (SASL)
RFC 2222, 1997
- Generic Security Service Application Program Interface (GSSAPI)

RFC 2078, 1997

- Java Cryptography Architecture
- Java Cryptography Extensions
- OpenSSH mit OpenSSL

Teil 1 erstellt unter Verwendung eines Seminarvortrags von Robert Schulz.

© Universität Mannheim, Rechenzentrum, 1998-2004.

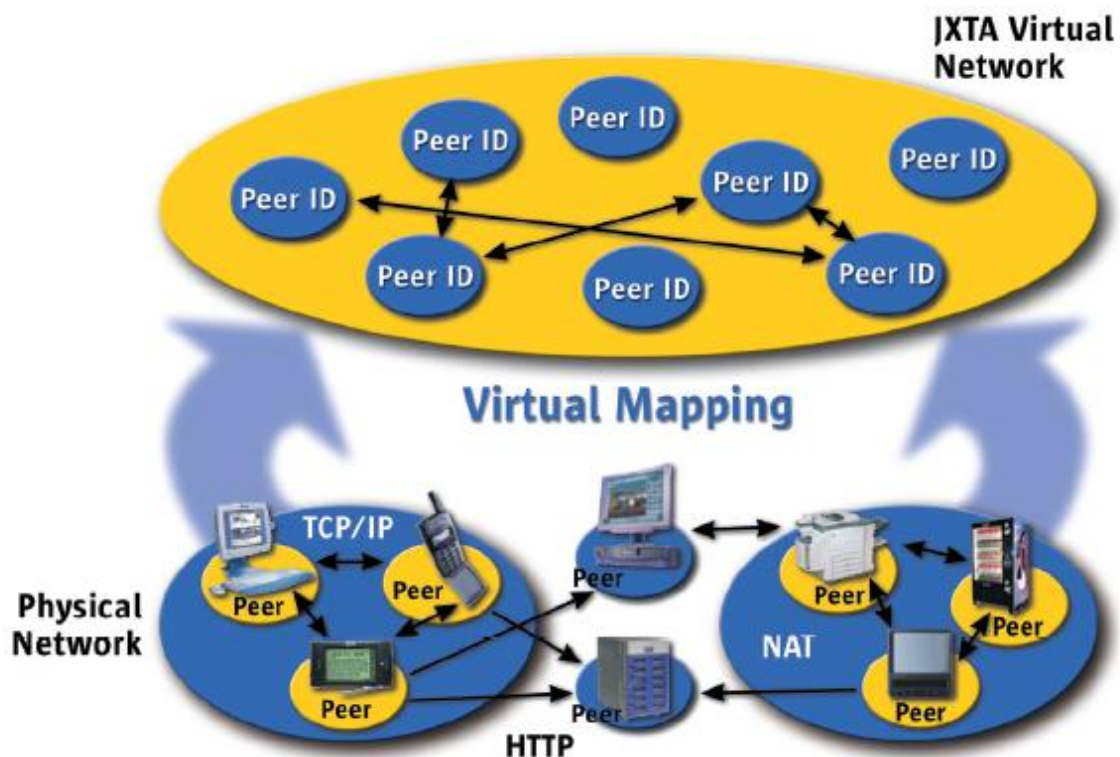
Last modified: Wed Jun 16 22:13:24 CEST 2004

24. Gnutella

- Einleitung P2P
- Gnutella

24.1. Einleitung P2P

- einfachste Form: Netz von Email-Partnern
- Peer-to-Peer: Netz zwischen 'Gleichgestellten' oder auch 'Gleichgesinnten'
- es gibt keine übergeordnete Instanz
wie etwa Zertifizierungsstelle (DFN), Web-Service Anbieter (web.de), Marktplatz Betreiber (eBay)
- jeder ist Anbieter und Abnehmer gleichzeitig
wie etwa bei Tauschbörsen
- verteiltes System zum Speichern und Finden von Informationen
- Technik:
virtuelles Netz auf Basis der Internet-Infrastruktur, mit eigenen (anonymen) Adressen (eindeutigen Identifikatoren) in einem eigenen Namensraum



P2P Netzwerke als virtuelle Netzwerke
Quelle: www.sun.com/jxta

- Einrichtung und Administration wesentlich einfacher
als die eines Domain-Name Servers und eines Web Servers

- Software hat gleichzeitig TCP/IP-Server und -Client Funktionen
- Kommunikation bei Bedarf über SSL/TLS, Schlüssel werden z.B. bei JXTA implizit angelegt und verwendet
- funktionieren meist auch durch Firewalls hindurch (Tunnel über HTTP, Port 80)
- Nachbarn im P2P-Netz können im Internet weit von einander entfernt sein
- wissenschaftlich interessante Suchverfahren, z.B. ähnlich wie verteilte Hashtabellen

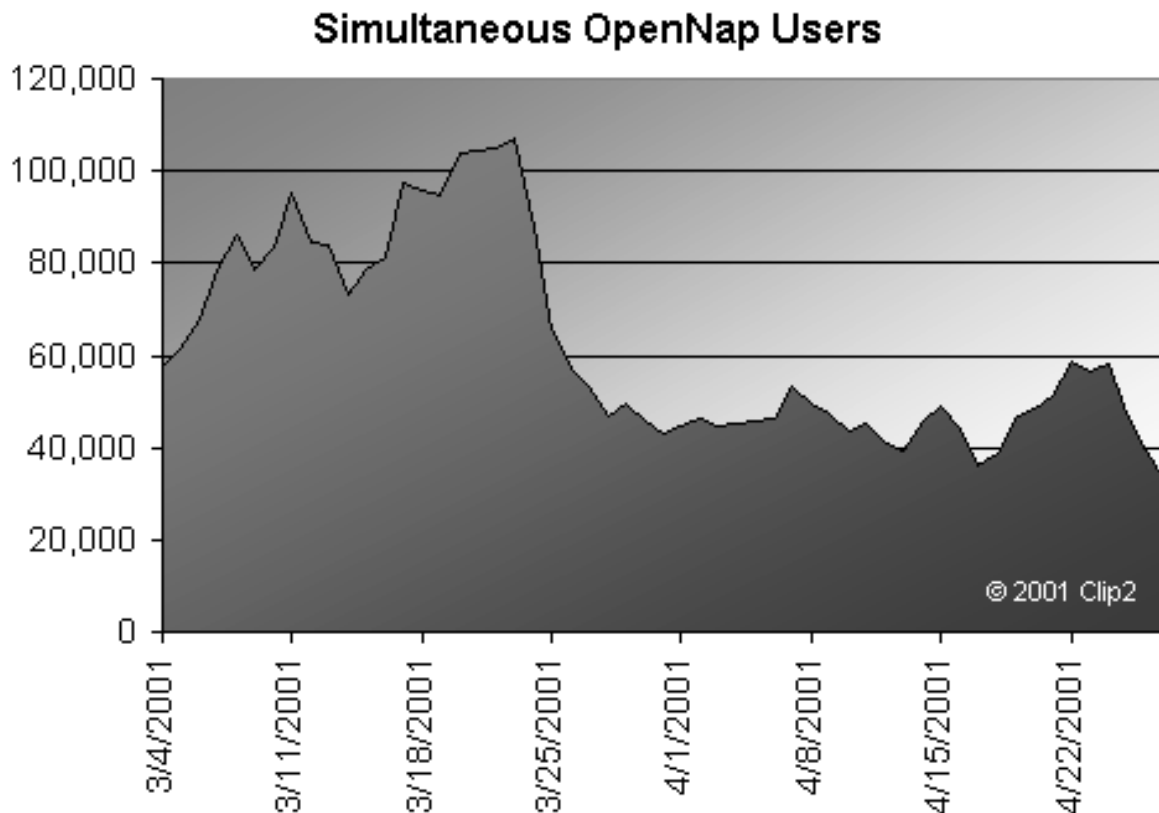
Formen des Peer-to-Peer

- *servergestütztes* Peer-to-Peer:
ein (oder replizierte) Server verwaltet ein Verzeichnis über die Teilnehmer oder die Angebote der Teilnehmer
z.B. Napster



Nachteil: zentraler Server kann ausfallen, überlastet sein, oder geschlossen werden

Vorteil: effiziente Suche durch Anfrage an Server



Quelle: Clip2, Kelly Truelove, OpenP2P.com

Rückgang der (Open) Napster User nach dem juristischen Einschreiten der RIAA. Die Anzahl der Server ist von ca. 200 auf ca.

100 zurückgegangen.

- *reines* Peer-to-Peer:
jeder Knoten des virtuellen Netzes ist gleich und kein Knoten hat einen Überblick (Verzeichnisse) über das Netz
z.B. Gnutella



Nachteil: Suchen schwer und aufwändig (Komplexität u.U. exponentiell)

Vorteil: kein Ausfall zentraler Dienste, schwer angreifbar (juristisch, technisch)

- *hybrides* Peer-to-Peer:
einige Knoten übernehmen Verzeichnisdienste für die anderen Knoten, sogenannte Superknoten, Funktionalität des P2P wird durch Ausfall eines oder mehrer Superknoten nicht beeinträchtigt, lediglich schlechtere Performance
z.B. KaZaA, JXTA



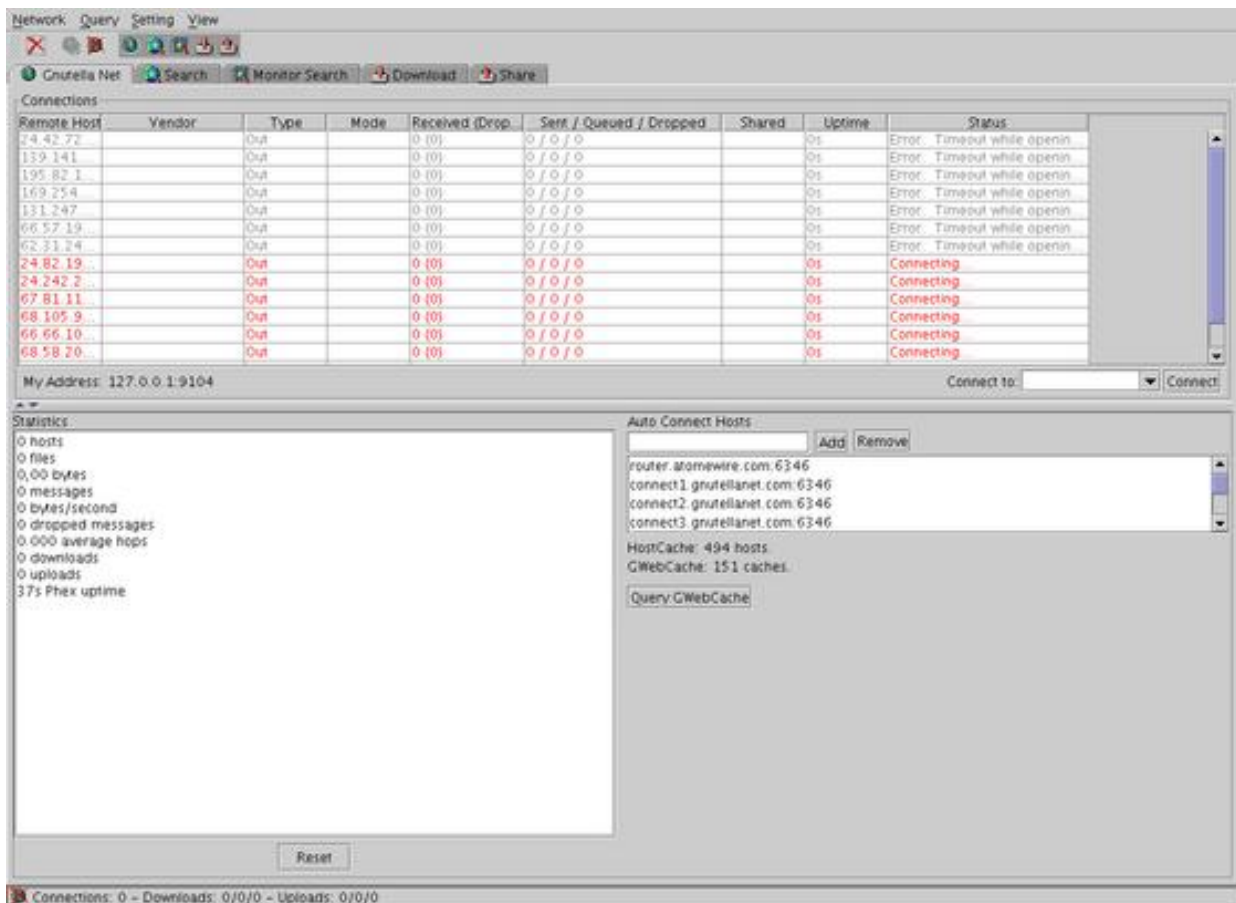
Nachteil: komplexere Software

Vorteil: einigermaßen effiziente Suche durch Anfrage an Superknoten, Ausfallsicherheit durch Wechsel / Übernahme der Superknoten

Software Architektur

- **Bootstrapping**
Generierung einer eindeutigen ID, Generierung eines Schlüsselpaares, Finden des Netzes
- **Management**
Einfügen und Entfernen von Knoten, Aufbau der virtuellen lokalen Netzumgebung, Zuordnung der virtuellen Adressen (IDs) zu IP-Adressen, Erkennen von 'verschwundenen' Nachbarn
- **Routing**
Finden der Wege zu entfernten Knoten im virtuellen (und im realen) Netz, Senden zu entfernten Knoten
- **Anwendungen**
Finden von Dateien, Senden und Empfangen von Dateien, oder Chat-Funktionen, Messaging, etc.

24.2. Gnutella



Phex Gnutella Bildschirm

- erste Entwickler J. Franklin & T. Pepper von Nullsoft
- im März 2000 auf [Slashdot](#) angekündigt
- Weiterentwicklung durch eine Gemeinschaft von Programmieren
z.T. durch Reengineering, da am Anfang keine Spezifikation vorhanden
- reines P2P, alles dezentral
- aktuelle Spezifikation 0.4, rev 1.2
- aktuelle Untersuchungen und Entwicklungen:
Verbesserungen des Routings, Recovery und der Queries
- verteiltes System zum Austausch von Dateien

Features

- die Knoten heißen *gnodes*
- jeder gnode ist faktisch immer Anbieter und Nachfrager von Dateien, d.h. die Client und die Server-Seite sind immer gleichzeitig aktiv
- das Netzwerk der gnodes besitzt keine Hierarchie, jeder gnode ist gleichberechtigt und bietet die gleiche Funktionalität
- Knoten, die höhere Bandbreite bieten, werden für Datei-Downloads bevorzugt
- die Knoten kennen nur ihre direkten Nachbarn, andere Knoten sind nur indirekt durch Antworten auf Anfragen bekannt
- dadurch wird ein hoher Grad von Anonymität erreicht
- Queries (Anfragen) werden zum nächsten Nachbarn gesendet, diese schicken sie dann weiter an alle ihnen bekannten

Nachbarn

- beim Ausfall eines Knotens versuchen sich diesen Nachbarn 'kurzzuschliessen'
- steuernde Parameter sind
die *Anzahl der gleichzeitigen Verbindungen* zu Nachbarknoten (ähnlich Game-of-Life)
und die *TTL der Nachrichten*
- die Knoten versuchen diese Parameter konstant zu halten,
hohe Verbindungsanzahlen bieten bessere Suchmöglichkeiten, aber auch eine höhere Netzlast
hohe TTL Zahlen bieten grössere Suchtiefe und auch eine höhere Netzlast

Gnutella Protokoll

Das Gnutella Protokoll zur Kommunikation ist erstaunlich einfach. Es besteht aus nur fünf (sieben) Kommandos / Nachrichtentypen / Descriptoren und nutzt zusätzlich zwei Funktionen des HTTP 1.0 Protokolls.

Bootstrapping

- Wie ein Knoten an die IP-Adressen von anderen gnodes kommt ist nicht definiert.
- Auch welche Ports verwendet werden sollen ist nicht definiert, oft Port 6346 am Ziel und Port 9104 lokal.
- Ein neuer Knoten baut eine TCP/IP Verbindung zu einem gnode auf.
- In der Verbindung sendet die Neue

```
GNUTELLA CONNECT/0.4\n\n
```

- Falls der gewünschte gnode antworten will, sendet er

```
GNUTELLA OK\n\n
```

- Danach können Gnutella Nachrichten ausgetauscht werden. Jede Nachrichten enthält folgende Felder im Descriptor:
- Gnode Identifikation (Gnode ID) des Senders (16 Byte) bei Ping und Query, des Fragenden bei Pong und QueryHit (in Spezifikation: Descriptor ID)
- Funktionscode (payload descriptor)
- TTL (time to live), wird bei jedem Schritt im Netzwerk um 1 decrementiert. Ist der Wert 0, wird die Nachricht verworfen.
- Hops, wird bei jedem Schritt im Netzwerk um 1 incrementiert
- Datenlänge (payload length), bestimmt, wo die Nachricht aufhört und eine neue anfängt.

Management

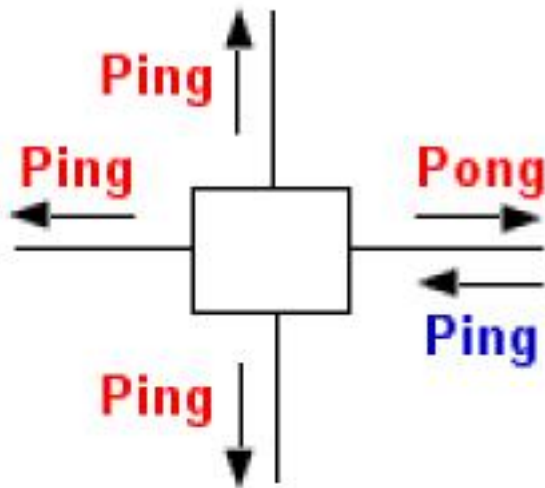
- Die Existenz anderer Knoten wird durch die *Ping* Nachricht erkundet. Ping enthält keine weiteren Informationen.
- Knoten, die eine Ping Nachricht empfangen können mit einer *Pong* Nachricht antworten. Es kann zusätzlich mit gespeicherten Pong Nachrichten anderer gnodes geantwortet werden. Die Antwort enthält folgende Felder:
- Port und IP-Adresse des Antwortenden
- Anzahl der Dateien, die öffentlich heruntergeladen werden können (shared files)
- Summe aller Kilobytes der verfügbaren Dateien.
- Die *Query* Nachricht dient dem Suchen im Gnutella Netzwerk (gehört eigentlich zur Anwendung). Die Anfrage enthält folgende Felder:
- kleinste Download-Bandbreite (in KB/sec), die der Anfrager akzeptieren will, z.B. 0 für jede beliebige Bandbreite
- Suchzeichenkette, meist Dateinamen, die Wildcards * enthalten dürfen.
- Die *QueryHit* Nachricht enthält die Antwort auf eine Query-Nachricht. Gnodes die keine passenden Dateien anbieten können generieren keine QueryHit Nachricht (gehört eigentlich zur Anwendung). Die Antwort enthält folgende Felder:
- Anzahl der Dateien, für die die Anfrage zutrifft
- Port und IP-Adresse des Antwortenden

- Download-Bandbreite (in KB/sec), die der Antwortende bieten kann ($\geq$ der verlangten)
- für jede Datei: einen Index, die Grösse in KB und ihren Namen
- den Gnode Identifier des Anbietenden (servent identifier)

Routing

Jeder Gnode hat nur Verbindungen zu seinen unmittelbaren Nachbarn.

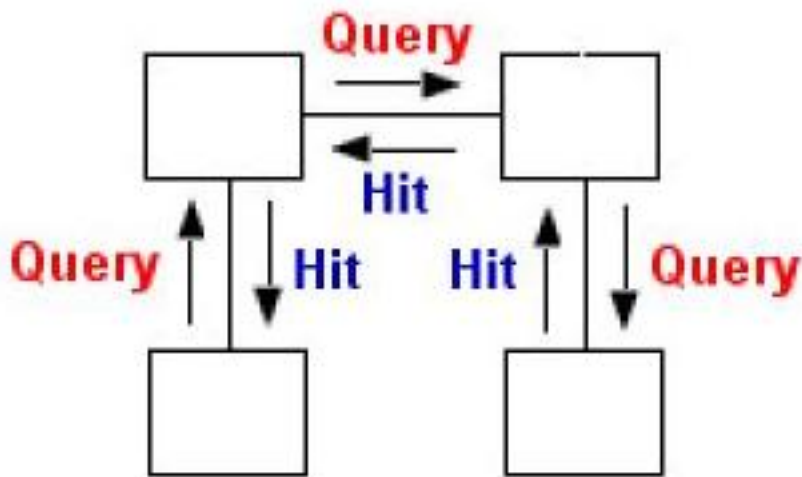
1. Für jede Nachricht wird das TTL Feld decrementiert und das Hops Feld hochgezählt. Ist das TTL Feld Null, wird die Nachricht verworfen.
2. Falls der Gnode erkennt, dass eine Nachricht innerhalb eines (kurzen) Zeitraums, erneut eingetroffen ist, wird die Nachricht verworfen. Dient zur Schleifenerkennung (loop detection).
3. Eingehende Ping und Query Nachrichten werden mit einer Pong bzw. einer QueryHit Nachricht über die eigene IP-Adresse bzw. der Liste der Treffer beantwortet.
4. Eingehende Ping und Query Nachrichten werden an alle Verbindungen (ausser der eingehenden) weiter geleitet falls das TTL Feld grösser als Null ist.



Gnutella Ping - Pong Routing

Quelle: Gnutella Spezifikation 0.4, <http://www.clip2.com/>

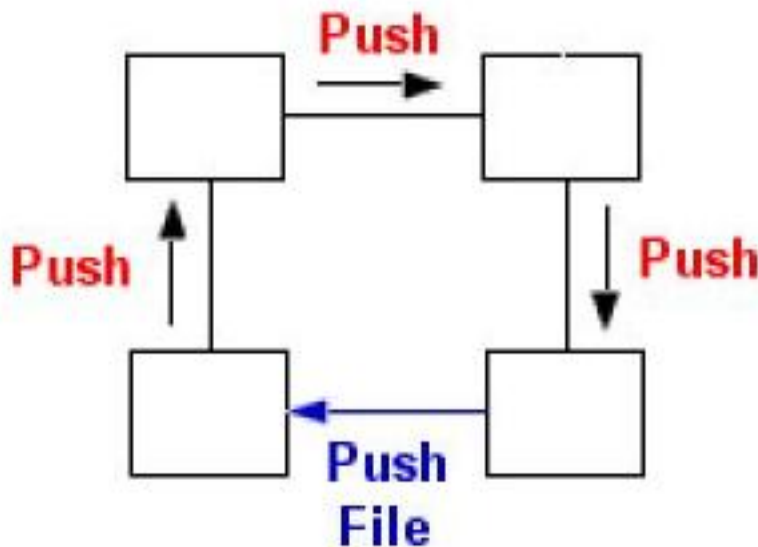
5. Eingehende Pong und QueryHit Nachrichten werden nur auf der Verbindung weiter geleitet, von der die entsprechenden Ping bzw. Query Nachrichten kamen, andernfalls werden sie verworfen. Dazu werden die Nachrichten gespeichert und über den Nachrichten Descriptor verglichen (evtl. wie Hash Tabelle).



Gnutella Query - QueryHit Routing

Quelle: Gnutella Spezifikation 0.4, <http://www.clip2.com/>

6. Push Nachrichten werden nur in der Verbindung weiter geleitet, von der eine entsprechende QueryHit Anfrage beantwortet wurde, andernfalls werden sie verworfen.



Gnutella Push Routing

Quelle: Gnutella Spezifikation 0.4, <http://www.clip2.com/>

Anwendungen

Zum Transport der Dateien bedient sich Gnutella des HTTP 1.0 Protokolls.

Die HTTP Verbindung wird unabhängig vom Gnutella Netzwerk direkt zwischen Client und Server (IP-Adresse und Port steht in QueryHit) aufgebaut.

- *HTTP GET* wird verwendet, um eine Datei, die durch eine QueryHit Antwort spezifiziert ist, herunterzuladen.

```
GET /get/<Index in QueryHit>/<DateiName>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-~\r\n
```

```
User-Agent: Gnutella\r\n\r\n
```

- Der Antwortende liefert die Datei entsprechend HTTP als *HTTP 200 OK* aus.

```
HTTP 200 OK\r\nServer: Gnutella\r\nContent-type: application/binary\r\nContent-length: xxx\r\n\r\n<xxx Bytes Dateinhalt>
```

- Die *Push* Nachricht dient dem Herunterladen von Dateien, wenn der Server sich hinter einer Firewall befindet (gehört eigentlich zum Management). Die Nachricht enthält folgende Felder:

- den Gnode Identifier des Anbietenden (servent identifier), hier also des Empfängers der Nachricht
- Port und IP-Adresse des Anfragenden
- Index der gewünschten Datei

Anders als bei HTTP GET, bei dem der Client die TCP/IP Verbindung zum Datei-Server aufbaut, baut hier der Datei-Server die Verbindung zum Client auf (falls möglich). Der Server schickt einen GIV Header, worauf der Client mit dem normalen HTTP GET Request weiter macht.

```
GIV <Index>:<Gnode-ID>/<DateiName>\r\nGET /get/<Index>/<DateiName>/ HTTP/1.0\r\n...\r\nHTTP 200 OK\r\n...
```

Erweiterungen des Protokolls

- Die QueryHit Nachricht wird u.A. um folgende Felder pro Datei erweitert.
- *Vendor Code*: zur Unterscheidung verschiedener Gnutella Implementierungen
- *flags*: zur Kennzeichnung von (be / über) lasteten Servern
- Felder, die weitere Eigenschaften der Dateien beschreiben, z.B. bei mp3-Dateien die Spieldauer oder Sample Rate

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Wed Jun 16 22:15:38 CEST 2004

25. JXTA

- Einleitung
 - Java Anwendungen
 - Protokolle
-

25.1. Einleitung

- JXTA von Juxtapose (nebeneinander, zusammen)
- Protokolle für P2P (in XML Schema)
- Nachrichten im XML-Format
- Einteilung der Peers in Gruppen: PeerGroups
- Kommunikation bei Bedarf verschlüsselt
- OpenSource, Lizenz ähnlich Apache
- Plattformunabhängig:
C/C++, Perl, Java, bzw. PC, PDA, bzw. TCP/IP, HTTP
- Referenzimplementierung in Java
- Core Protokolle:
Peers, Advertisements, Gruppen, Pipes, Sicherheit, Monitoring
müssen implementiert sein
- Service Protokolle:
Dateienaustausch, Nachrichtenaustausch, Indizieren und Suchen, PKI, Ressourcen Reservierung
können implementiert sein
- Anwendungsprotokolle:
Chat, Content Management, verteilte Zusammenarbeit, Messaging verteilte Auktionen

Begriffe und Konzepte

Peers

wie in anderen P2P Systemen, die elementaren (Software) Einheiten, oft identisch mit Geräten im Netzwerk

Peer Groups

Peers **müssen** in Peer-Gruppen enthalten sein. Ein Peer kann in mehreren Gruppen gleichzeitig sein. Ein Peer ist immer in der World Peer Group und in der Net Peer Group. Alle Services beziehen sich nur auf die aktive Gruppe.

Die Gruppen bieten einen Namensraum und eine Sicherheitsumgebung für den Zugang und die Kommunikation.

Pipes

elementare Kommunikationskanäle, vergleichbar zu (UDP) Sockets.

Implementiert z.B. mit TCP/IP oder HTTP, in der einfachsten Form asynchron und unzuverlässig.

Advertisements

allgemeiner Publikationsmechanismus. XML Dokumente, die alle JXTA Ressourcen (Peers, Peer Groups, Pipes, etc.) bekannt machen. Advertisements werden meist lokal gecached.

Discovery

Basismechanismus zum Finden von Advertisements, also zum Finden aller JXTA Ressourcen.

Modules

Abstraktion für Programmcode. z.B. Java Class oder Jar, DLL, SO. Ermöglicht es einem Peer neue Dienste zu erzeugen, in dem er Code sucht, findet, lädt und ausführt.

Network Services

Dienste, die von einem Peer oder einer Gruppe von Peers (evtl. kooperativ) angeboten werden.

Messages

elementare Einheit der kommunizierten Nachrichten. Können im XML- oder Binärformat vorliegen. Eine Nachricht ist eine geordnete Folge von Nachrichtenelementen, die einen Typ und einen Namen haben.

Security

Die JXTA XML Nachrichten können Informationen über Zertifikate, Credentials, Digests, Public Keys etc. enthalten. Die Nachrichtenelemente können verschlüsselt und / oder signiert sein. Jeder JXTA Peer besitzt einen Public und Private Key.

IDs

Eindeutiger Kennzeichner einer JXTA Resource (Peer, Peer Group, Pipes, Services, etc.). IDs sind wie URNs aufgebaut:
urn:jxta:uuid-2233...000203

25.2. Java Anwendungen

Hello World mit Java JXTA

1. zum Compilieren und Ausführen werden folgende Jar-Dateien benötigt (z.B. aus /opt/JXTA_Demo/lib):

```
jxta.jar
log4j.jar
jxtasecurity.jar
jxtaptls.jar
minimalBC.jar
beepcore.jar
cryptix-asn1.jar
cryptix32.jar
```

2. `PeerGroupFactory.newNetPeerGroup()` liefert ein Objekt, das die NetPeer Gruppe repräsentiert und initialisiert das JXTA Laufzeitsystem
3. über diese `PeerGroup` erfolgen alle weiteren Kontakte mit der JXTA Umgebung
4. `getPeerGroupID()` zeigt die ID der (Net) Peer Group
5. `getPeerID()` liefert die eigene ID des Peers
6. `getPeerName()` zeigt den eigenen Namen
7. nach der Initialisierung der JXTA Umgebung ist der Peer dann in der Net Peer Group sichtbar
8. `getPeerGroupAdvertisement()` liefert das Advertisement der Net Peer Gruppe, das anschliessend ausgegeben wird

```
import net.jxta.peergroup.PeerGroup;
import net.jxta.peergroup.PeerGroupID;
import net.jxta.peergroup.PeerGroupFactory;

import net.jxta.peer.PeerID;
import net.jxta.protocol.PeerGroupAdvertisement;
import net.jxta.exception.PeerGroupException;
```

```

import net.jxta.discovery.DiscoveryService;
import net.jxta.pipe.PipeService;
import net.jxta.membership.MembershipService;
import net.jxta.resolver.ResolverService;

import net.jxta.document.StructuredTextDocument;
import net.jxta.document.MimeMediaType;

import java.io.IOException;

public class HelloWorldJXTA {

    PeerGroup gruppe = null;
    Screen sc = new Screen();

    public static void main(String[] args) {
        HelloWorldJXTA hello = new HelloWorldJXTA();
        hello.startJXTA();
    }

    void startJXTA() {
        sc.println("starting JXTA hello world");
        try {
            gruppe = PeerGroupFactory.newNetPeerGroup();
        } catch (PeerGroupException e) {
            e.printStackTrace();
            return;
        }
        if ( gruppe == null ) {
            sc.println("Net Group not found, no way to continue");
            return;
        }

        sc.println("Net Group is: " + gruppe);
        PeerGroupID pgid = gruppe.getPeerGroupID();
        sc.println("pgid: " + pgid);
        PeerID pid = gruppe.getPeerID();
        sc.println("pid: " + pid);
        String pname = gruppe.getPeerName();
        sc.println("pname: " + pname);

        PeerGroupAdvertisement pgadv = gruppe.getPeerGroupAdvertisement();
        sc.println("pgadv: ");
        StructuredTextDocument doc = (StructuredTextDocument)
            pgadv.getDocument( new MimeMediaType("text/xml") );
        try {
            doc.sendToWriter( sc );
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            sc.flush();
        }

        // core services:
        DiscoveryService disco = gruppe.getDiscoveryService();
        PipeService pipe = gruppe.getPipeService();
        MembershipService member = gruppe.getMembershipService();
        ResolverService resolv = gruppe.getResolverService();
        sc.println("got core services");
    }
}

```

Die Ausgabe des Programms zeigt der Reihe nach die Peer Group ID, die Peer ID und den Peer Namen:

```

starting JXTA hello world
Net Group is: net.jxta.impl.peergroup.PeerGroupInterface@118958e
pgid: urn:jxta:jxta-NetGroup
pid: urn:jxta:uuid-59616261646162614A78746150325033499E205EAB4C4933BF588B24A8CA76C703
pname: intec peer

```

Die weitere Ausgabe zeigt dann das XML Dokument, das das Advertisement der Net Peer Group enthält:

```
pgadv:
<?xml version="1.0"?>

<!DOCTYPE jxta:PGA>

<jxta:PGA xmlns:jxta="http://jxta.org">
  <GID>
    urn:jxta:jxta-NetGroup
  </GID>
  <MSID>
    urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206
  </MSID>
  <Name>
    NetPeerGroup
  </Name>
  <Desc>
    NetPeerGroup by default
  </Desc>
</jxta:PGA>
```

Der Rest des Programms zeigt noch den Zugriff auf verschiedene Services, die von der Gruppe angeboten werden.

JXTA Konfigurator

Beim ersten Ausführen des obigen (oder eines beliebigen) JXTA Programms wird ein Konfigurationszyklus durchlaufen. Dabei werden die Netzwerkeinstellungen, Namen und Passwörter abgefragt.

See "<http://shell.jxta.org/index.html>" for config help

basic | advanced | Rendezvous/Relays | Security

Basic settings

Peer Name (Mandatory)

☐ Use proxy server (if behind firewall)

Proxy address

OK Cancel

JXTA Konfigurator

- der *Peer Name* ist der Name, unter dem der Peer in den Peer Gruppen erscheint
- unter *Use Proxy Server* kann ggf. ein Proxy eingerichtet werden, mit dem man über eine Firewall hinwegkommt
- auf der *Advanced* Seite sind die Netzwerk Parameter einzustellen
- *Trace Level* bietet bei Bedarf mehr Informationen über den Ablauf. Die Level entsprechen den Levels von Log4j.
- *HTTP-Settings*:
muss man deaktivieren, wenn man zu Hause in einem privaten Netz testet
- *TCP-Settings*:
testet man mehrere Peers auf einem Rechner benötigt jeder einen eigenen Port.
hat man mehrere Netzwerkinterfaces ist ggf. eine IP-Adresse einzutragen
- in der Netzwerkkonfiguration des Rechners muss auf jeden Fall eine default Route eingetragen sein, damit Multicast für Peer Discovery funktioniert.
- unter *Rendezvous/Relays* sind zunächst keine Eingaben notwendig
- auf der *Security* Seite wird ein Name und ein Passwort für die privaten Schlüssel erfragt (der Name tritt nicht nach aussen)

in Erscheinung)

- der Konfigurator erstellt anschliessend ein Verzeichnis `.jxta` in dem alle Konfigurationsinformationen und die privaten Schlüssel abgelegt werden
- existiert in dem Verzeichnis `.jxta` eine Datei `reconf` wird der Konfigurationszyklus erneut durchlaufen
- wird das Verzeichnis `.jxta` gelöscht wird es erneut angelegt. Dabei werden neue Schlüssel erzeugt.

Wird das Programm erneut ausgeführt, werden nur noch der Namen und das Passwort für den privaten Schlüssel abgefragt.



Name und Passwort für den privaten Schlüssel

Konfigurationsdateien

Das Verzeichnis `.jxta` enthält folgende Teile:

- die Datei `PlatformConfig` enthält die meisten Informationen des Konfigurators
- das Verzeichnis `cm` enthält gecachte Advertisements, z.B. für alle bekannten Peer Gruppen; für jede Peer Gruppe gibt es ein eigenes Verzeichnis, das wiederum Unterverzeichnisse für Peers, PeerGruppen etc. enthält.
bei JXTA 2.0 gibt es dafür `Xindice` Dateien
- das Verzeichnis `pse` enthält den privaten und öffentlichen Schlüssel, die erforderlichen Passwörter und die Schlüssel der Root-CA (ist der Peer selbst)

JXTA Shell

Für die ersten Schritte mit JXTA existiert eine (Unix ähnliche) Shell. Diese Shell bietet für alle wichtigen JXTA Funktionen kleine Kommandos an. Damit lässt sich z.B. ein Überblick über die Peer Gruppen, Peers, Services etc gewinnen.

```
===== Welcome to the JXTAShell Version 1.0 =====
```

The JXTA Shell provides an interactive environment to the JXTA platform. The Shell provides basic commands to discover peers and peer groups, to join and resign from peer groups, to create pipes between peers, and to send pipe messages. The Shell provides environment variables that permit binding symbolic names to Jxta platform objects. Environment variables allow Shell commands to exchange data between themselves. The shell command 'env' displays all defined environment variables in the current Shell session.

The Shell creates a Jxta InputPipe (stdin) for reading input from the keyboard, and a Jxta OutputPipe (stdout) to display information on the Shell console. All commands executed by the Shell have their initial 'stdin' and 'stdout' set up to the Shell's stdin and stdout pipes. The Shell also creates the environment variable 'stdgroup' that contains the current JXTA PeerGroup in which the Shell and commands are executed.

A new Shell can be forked within a Shell. The 'Shell -s' command starts a new Shell with a new Shell window. The Shell can also read a command script file via the 'Shell -f myfile'.

A 'man' command is available to list the commands available. Type 'man <command>' to get help about a particular command. To exit the Shell, use the 'exit' command.

```
JXTA>peers
peer0: name = heinz
JXTA>peers -r
peer discovery message sent
JXTA>peers
peer0: name = heinz
peer1: name = intec peer
JXTA\
```

Shell zum Testen von JXTA Anwendungen

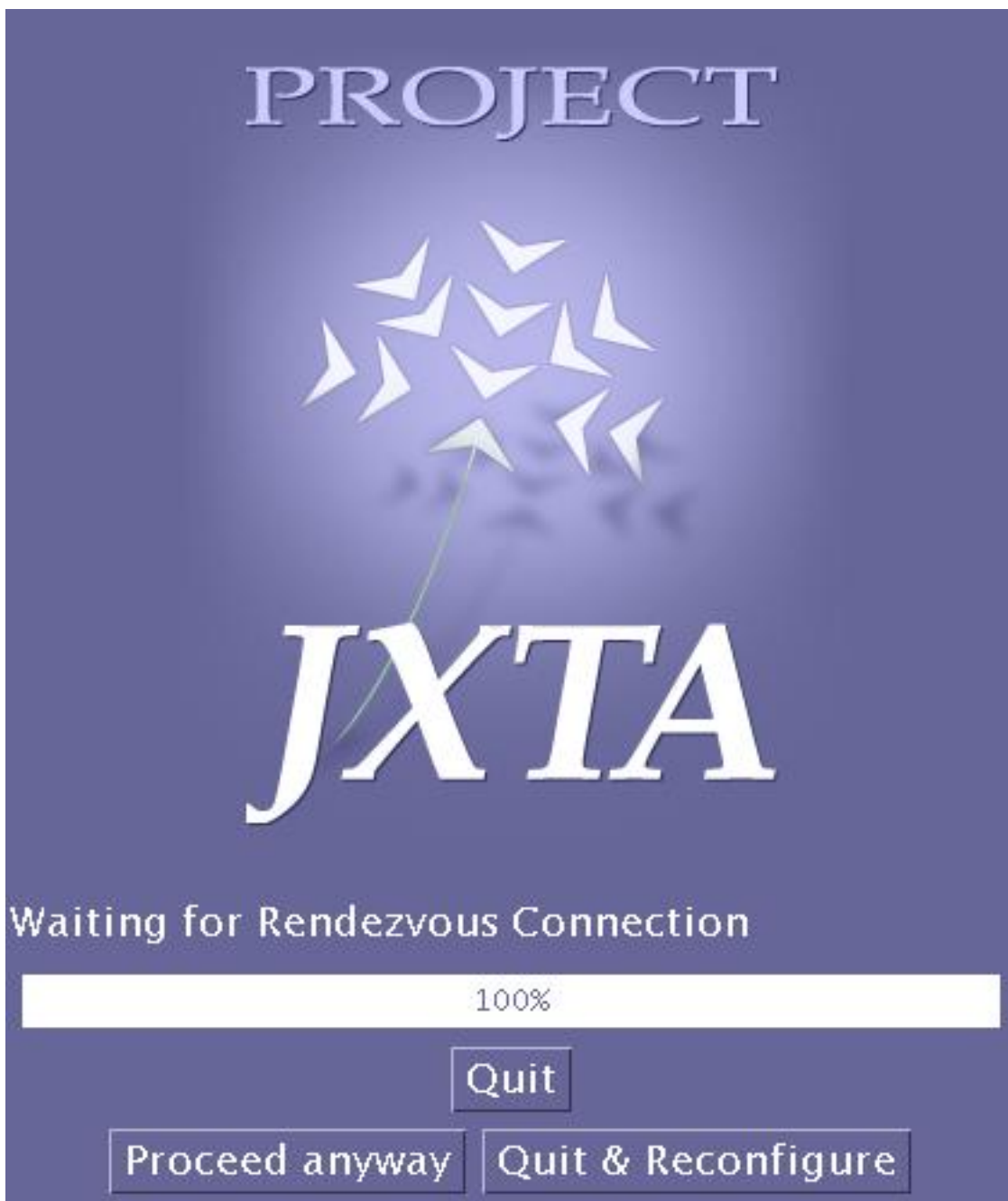
Übersicht über die Shell Kommandos, wie sie vom Kommando man geliefert wird:

cat	Concatenate and display a Shell object
chpgrp	Change the current peer group
clear	Clear the shell's screen
cms	No description available for this ShellApp
env	Display environment variable
exit	Exit the Shell
exportfile	Export to an external file
get	Get data from a pipe message
grep	Search for matching patterns
groups	Discover peer groups
help	No description available for this ShellApp
history	No description available for this ShellApp
importfile	Import an external file
instjar	Installs jar-files containing additional Shell commands
join	Join a peer group
leave	Leave a peer group
man	An on-line help command that displays information about a specific Shell command
mkadv	Make an advertisement
mkmsg	Make a pipe message
mkpgrp	Create a new peer group
mkpipe	Create a pipe

more	Page through a Shell object
peerconfig	Peer Configuration
peerinfo	Get information about peers
peers	Discover peers
put	Put data into a pipe message
rdvserver	No description available for this ShellApp
rdvstatus	Display information about rendezvous
recv	Receive a message from a pipe
rshd	JXTA Shell command interpreter
search	Discover jxta advertisements
send	Send a message into a pipe
set	Set an environment variable
setenv	Set an environment variable
sftp	Send a file to another peer
share	Share an advertisement
Shell	JXTA Shell command interpreter
sql	Issue an SQL command (not implemented)
sqlshell	JXTA SQL Shell command interpreter
talk	Talk to another peer
transports	Display information about the message transports in the current group
uninstjar	Uninstalls jar-files previously installed with 'instjar'
version	No description available for this ShellApp
wc	Count the number of lines, words, and chars in an object
who	Display credential information
whoami	Display information about a peer or peergroup

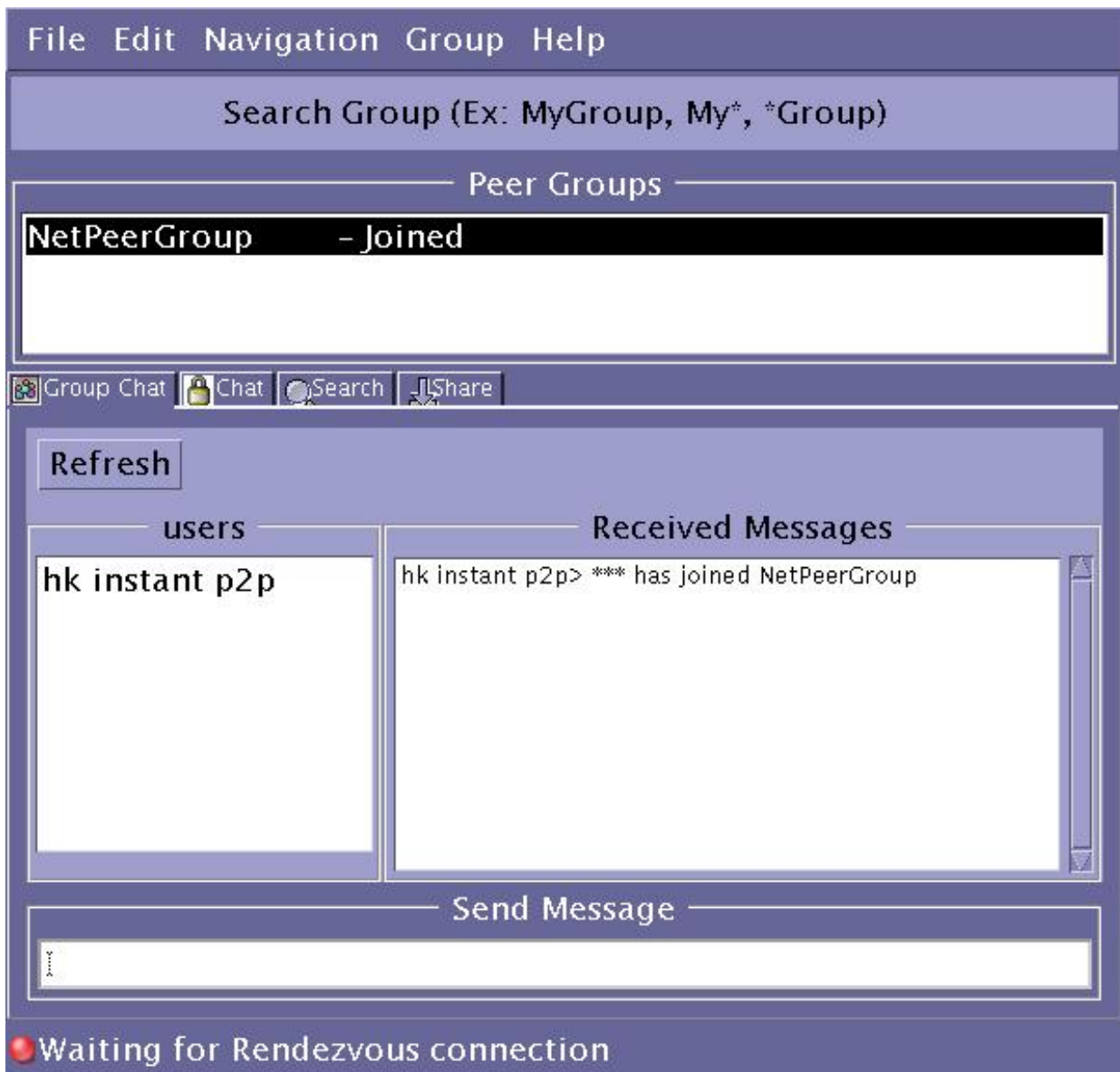
Instant JXTA Anwendung

Neben der JXTA Shell gibt es eine schöne Anwendung von JXTA, die Chat-Funktionen, Gruppen-Management und File-Sharing/Search erlaubt.



Begrüßungsbild der Instant JXTA Anwendung

Für die Konfiguration startet sich wieder der bekannte JXTA Konfigurator. Mit "Proceed Anyway" erreicht man den Hauptschirm der Anwendung.



Hauptschirm der Instant JXTA Anwendung

In der oberen Anzeige sind alle bekannten Peer Gruppen angezeigt. Ein "joined" hinter einem Gruppennamen gib an, dass man Mitglied dieser Gruppe ist. Die aktuelle Gruppe kann durch anklicken mit der Maus gewechselt werden. Die untere Anzeige hängt von der gewählten Funktion ab.



New Peer Group Name

☐ Act as rendezvous

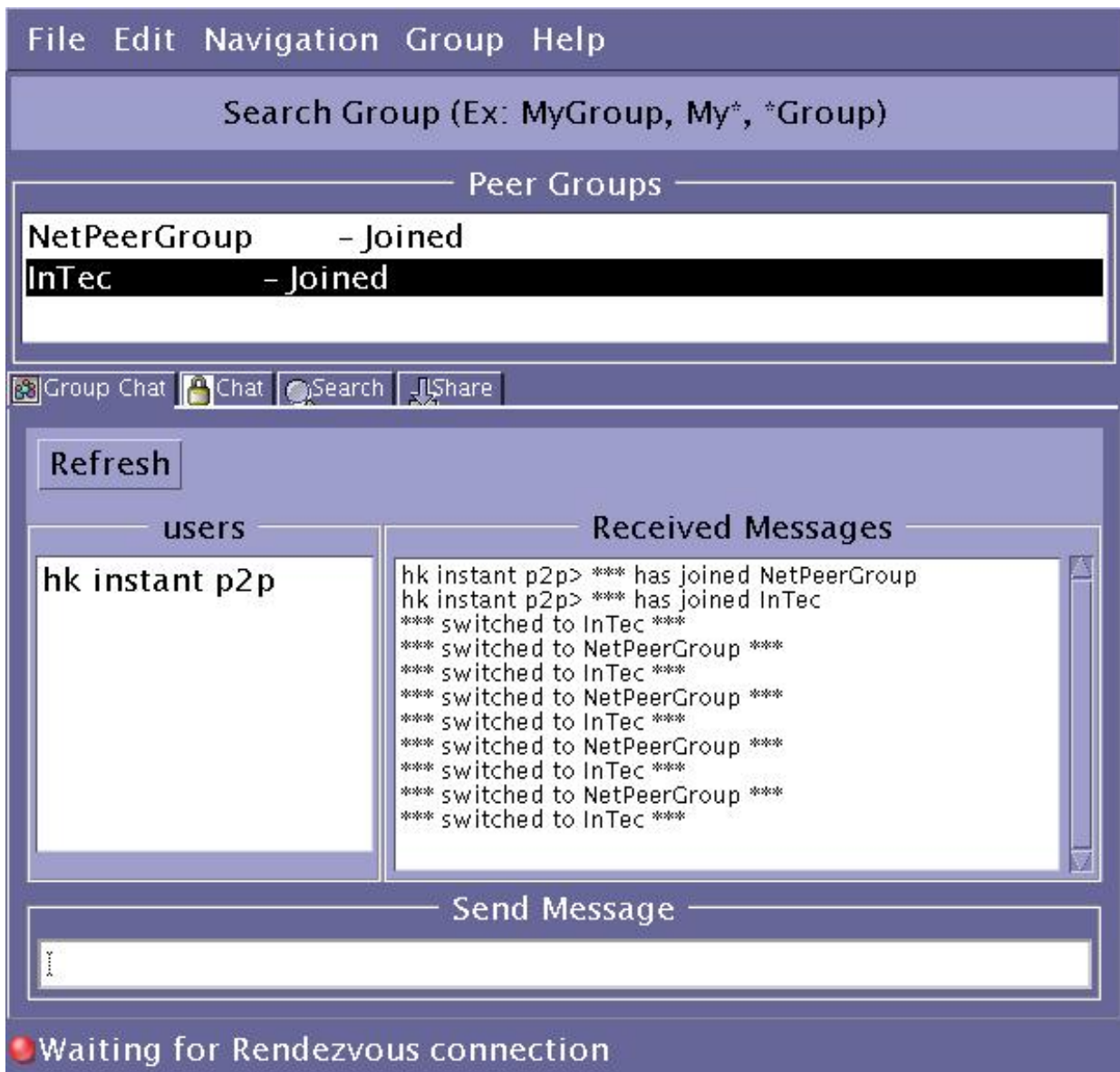
☐ Create Private Group

Group Password

Verify Password

Anlegen einer neuen Peer Gruppe mit Instant JXTA

Über das Menue "Group - New Group" kann eine neue Gruppe angelegt werden. Optional kann ein Passwort für den Zugang zur Gruppe festgelegt werden.



Hauptschirm der Instant JXTA Anwendung mit einer neuen Peer Gruppe "InTec"

Peer Groups finden und erzeugen

Peer Groups können über den Discovery Service gesucht werden. Dieser findet lokale oder entfernte Advertisements mit den Methoden `getLocalAdvertisements()` und `getRemoteAdvertisements()`. Soll eine bestimmte Peer Group verwendet werden, müssen alle Programme die richtige ID der Peer Group verwenden. Zur Erzeugung einer Peer Group dient die Methode `newGroup(Advertisement)` bzw. `newGroup( ID, Impl, name, desc)`.

```
String pgroupname = "InTecPeerGroup";
String pgURL = "jxta:uuid-3E2406ADF2E542FD87F1A9744052C30D02";

private PeerGroup searchPG(String pgroupname) throws Exception {
    PeerGroup ipg = null;
    DiscoveryService disco = gruppe.getDiscoveryService();
    Enumeration pgs = null;
    for (int i = RETRIES; i > 0; i-- ){
```

```

        try {
            pgs = disco.getLocalAdvertisements(
                DiscoveryService.GROUP,
                "Name",
                pgname);
            if ( pgs != null && pgs.hasMoreElements() ) break;
            disco.getRemoteAdvertisements(
                null,
                DiscoveryService.GROUP,
                "Name",
                pgname,
                1,
                null);

            try {
                Thread.sleep( TIMEOUT );
            } catch (InterruptedException e) { }
        } catch (IOException e) { }
    }

    PeerGroupAdvertisement pgAdv = null;
    if ( pgs != null && pgs.hasMoreElements() ) {
        sc.println("PeerGroup " + pgname + " found");
        try {
            pgAdv = (PeerGroupAdvertisement) pgs.nextElement();
            ipg = gruppe.newGroup( pgAdv );
        } catch (PeerGroupException e) {
            e.printStackTrace();
        }
        return ipg;
    }

    sc.println("PeerGroup " + pgname + " not found, creating new one");
    ModuleImplAdvertisement implAdv =
        gruppe.getAllPurposePeerGroupImplAdvertisement();

    ipg = gruppe.newGroup(
        mkGroupID(),
        implAdv,
        pgname,
        "The " + pgname + " Description"
    );

    return ipg;
}

private PeerGroupID mkGroupID() throws Exception {
    URL u = new URL( "urn", "", pgURL );
    return (PeerGroupID) IDFactory.fromURL( u );
}

```

Die Ausgabe könnte z.B. so aussehen:

```

searching InTecPeerGroup
PeerGroup InTecPeerGroup found
searched InTecPeerGroup
ipgid: urn:jxta:uuid-3E2406ADF2E542FD87F1A9744052C30D02

```

Mitglied einer Peer Group werden

Advertisements werden auch mit Hilfe des Discovery Services publiziert. Die entscheidende Methode ist `remotePublish()`.

```

PeerGroupAdvertisement adv = ipg.getPeerGroupAdvertisement();
DiscoveryService disco = gruppe.getDiscoveryService();
disco.remotePublish(adv, DiscoveryService.GROUP);
sc.println(pgname + " published");

```

Mit Hilfe des Membership Services kann man Mitglied in einer Peer Group werden. Der Service bietet dafür die Methoden `apply()` und `join()`.

```
private void joinPeerGroup(PeerGroup pg) throws Exception {
    MembershipService ms = pg.getMembershipService();

    AuthenticationCredential ac =
        new AuthenticationCredential(pg, null, null);
    Authenticator a = ms.apply( ac );

    if ( a.isReadyForJoin() ) {
        ms.join( a );
    } else {
        sc.println("a not ready for join "+ pg.getPeerGroupName());
    }
}
```

Beispiel Ausgabe:

```
joining peer group InTecPeerGroup
peer group joined
```

Peers finden

Wie bei Peer Groups findet man Peers auch mit Hilfe des Discovery Service. Die Methoden sind wieder `getLocalAdvertisements()` und `getRemoteAdvertisements()`, wobei jetzt der Typ `disco.PEER` verwendet wird.

```
private void discoverPeers(PeerGroup pg) throws Exception {
    sc.println("discovering peers in group " + pg.getPeerGroupName() );
    DiscoveryService disco = pg.getDiscoveryService();
    Enumeration ps = null;
    for (int i = RETRIES; i > 0; i-- ){
        try {
            disco.getRemoteAdvertisements(
                null, disco.PEER, null, null,2, null);
            try {
                Thread.sleep( TIMEOUT );
            } catch (InterruptedException e) { }
        } catch (IOException e) { }
    }
    ps = disco.getLocalAdvertisements(disco.PEER, null, null);
    // sc.println("peers: " + ps);
    if ( ps == null ) return;
    while ( ps.hasMoreElements() ) {
        try {
            PeerAdvertisement pa =
                (PeerAdvertisement) ps.nextElement();
            sc.println("found peer: " + pa.getName());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

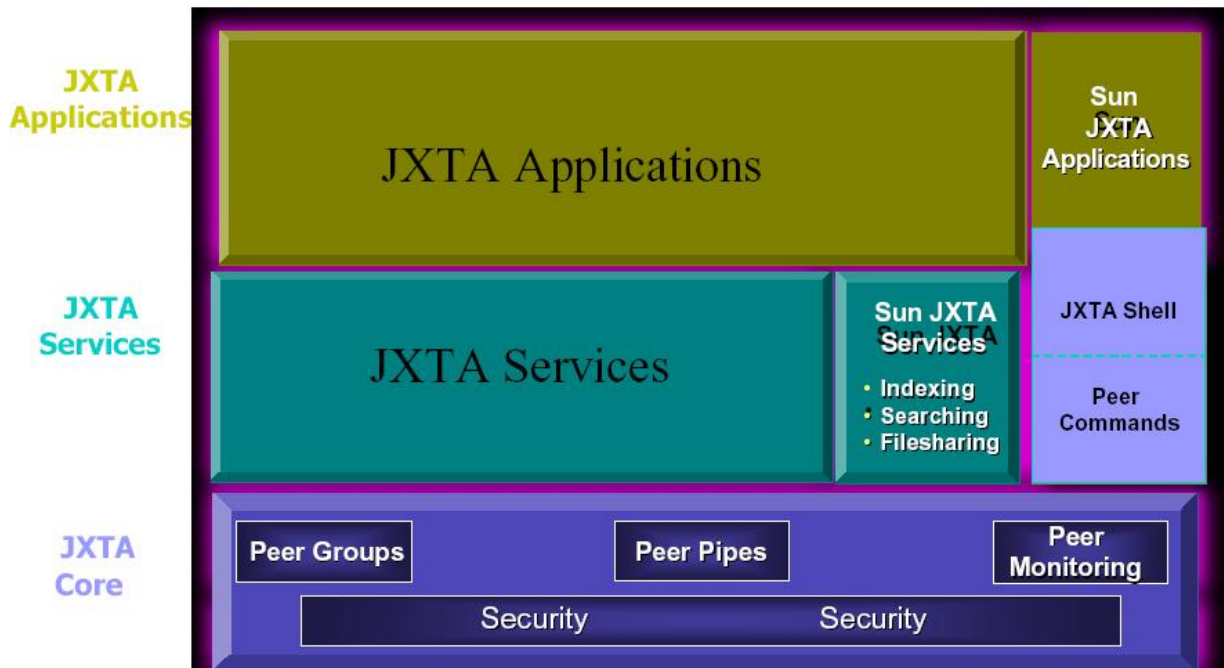
Die Ausgabe könnte wie folgt aussehen:

```
discovering peers
discovering peers in group InTecPeerGroup
found peer: InTec Peer
found peer: Heinz jxta_2.1 shell
found peer: InTec java Peer
discovering peers in group NetPeerGroup
found peer: InTec Peer
found peer: Heinz jxta_2.1 shell
found peer: InTec java Peer
peers discovered
```

Die weiteren Möglichkeiten von JXTA, wie die Verwendung von Pipes oder der Aufbau von verschlüsselten Verbindungen können hier nicht behandelt werden.

25.3. Protokolle

- Version 1.0 vom April 2001
- Version 2.0 vom Februar 2003
 - + Shared Resource Distributed Index (SRDI),
 - chaching of Advertisements,
 - + indexing of Advertisements mit Xindice



Schichten der JXTA Software-Architektur

Quelle: www.jxta.org

JXTA kann mit unidirektionalen, asynchronen Netzverbindungen arbeiten und eignet sich dadurch auch für PDAs und Handys, optimal sind natürlich TCP/IP oder HTTP.

Da Peers zu jeder Zeit kommen und wieder verschwinden können, dürfen in JXTA Programmen keine Annahmen über Antwortzeiten machen, bzw. nicht voraussetzen, dass überhaupt eine Antwort zurückkommt.

Content (also Dateien, MP3s, etc.) kann nach dem Publizieren unerreichbar sein (weil der Peer oder ein Routing-Peer verschwindet). Content kann zu mehreren Peers kopiert (repliziert) werden, wo er evtl. nicht mehr gelöscht werden kann.

Software Architektur

Einordnung der JXTA Protokolle in die P2P Software Architektur

- **Bootstrapping**
JXTA IDs, JXTA Configurator, JXTA Transport über TCP/IP, IP-Multicast, HTTP, TLS, ERTp
- **Management**
JXTA Advertisements, End Point Adresses, Private Security Environment (PSE), Cache Management (CM), Shared Resource Distributed Index (SRDI)
- **Routing**
End Point Routing Protocol, Peer Resolver Protocol, Resolver Service, Rendezvous Protocol

- **Anwendungen**

Peer Discovery, Pipe Binding, Peer Information, File Sharing, Chat-Funktionen über Pipes

IDs

JXTA IDs kann man als virtuelle IP-Adressen (mit Port-Nummern) auffassen. IDs gibt es zur Zeit für folgende Ressourcen:

- Peer-Gruppen
- Peers
- Pipes
- Content (Codats)
- Module-Klassen
- Module-Spezifikationen

Eigenschaften der IDs:

- eine ID muss die Resource vollständig referenzieren
- eine ID darf nur eine Resource referenzieren
- gleiche IDs sollen die gleiche Resource referenzieren, eine Resource soll nur eine ID haben
- der Inhalt oder Aufbau einer ID soll für Anwendungen unerheblich sein (opaque)

Allgemeiner Aufbau einer ID:

```
urn:jxta:format-1234567890abcde...
```

urn: und jxta: müssen als solche Zeichenketten vorkommen.

Für format gibt es z.Z. folgende Möglichkeiten: uuid für eindeutige IDs und jxta für vordefinierte IDs.

```
urn:jxta:uuid-123456789002
urn:jxta:uuid-123456789003

urn:jxta:jxta-WorldGroup
urn:jxta:jxta-NetGroup
urn:jxta:jxta-Null
```

Die letzten beiden Hexziffern einer uuid definieren den Typ der ID. Z.B. '02' für Peergruppen-IDs, '03' für Peer-IDs, '05' für Module-Class-IDs, '06' für Module-Service-IDs.

XML Schema Definitionen für JXTA IDs:

```
<xs:simpleType name="JXTAID">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="([uU][rR][nN]:[jJ][xX][tT][aA]:)+\-/>
  </xs:restriction>
</xs:simpleType>
```

Advertisements

Advertisements beschreiben JXTA Ressourcen im XML-Format. Die verschiedenen Advertisements werden durch XML Schema spezifiziert.

Die Definition für ein Peer Advertisement ist wie folgt:

```
<xs:element name="PA" type="jxta:PA"/>
<xs:complexType name="PA">
```

```

<xs:sequence>
  <xs:element name="PID" type="JXTAID"/>
  <xs:element name="GID" type="JXTAID"/>
  <xs:element name="Name" type="xs:string"
    minOccurs="0"/>
  <xs:element name="Desc" type="xs:anyType"
    minOccurs="0"/>
  <xs:element name="Svc" type="jxta:serviceParam"
    minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="serviceParam">
  <xs:sequence>
    <xs:element name="MCID" type="jxta:JXTAID"/>
    <xs:element name="Parm" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>

```

Beispiel für das Advertisement eines Peers:

```

<?xml version="1.0"?>
<!DOCTYPE jxta:PA>

<jxta:PA xmlns:jxta="http://jxta.org">
  <PID>
urn:jxta:uuid-iiiiiiiiiiiiiiiiiiii03
  </PID>
  <GID>
urn:jxta:jxta-WorldGroup
  </GID>
  <Name>
intec peer
  </Name>
  <Svc>
    <MCID>
urn:jxta:uuid-xxxxxxxxxxxxxxxxxxxx05
    </MCID>
    <Parm>
      <Addr>
tcp://10.1.1.254:9703/
      </Addr>
      <Addr>
jxtatls://uuid-iiiiiiiiiiiiiiiiiiii03/TlsTransport/jxta-WorldGroup
      </Addr>
      <Addr>
jxta://uuid-iiiiiiiiiiiiiiiiiiii03/
      </Addr>
    </Parm>
  </Svc>
  ...
</jxta:PA>

```

Die Definition für ein Peer Group Advertisement ist wie folgt:

```

<xs:element name="PGA" type="jxta:PGA"/>

<xs:complexType name="PGA">
  <xs:sequence>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="MSID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string"
      minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType"
      minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParam"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

Beispiel für das Advertisement einer Peergroup:

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta="http://jxta.org">
  <GID>
    urn:jxta:jxta-NetGroup
  </GID>
  <MSID>
    urn:jxta:uuid-mmmmmmmmmmmmmmmmmmm06
  </MSID>
  <Name>
    NetPeerGroup
  </Name>
  <Desc>
    NetPeerGroup by default
  </Desc>
</jxta:PGA>
```

Peer Resolver Protocol (PRP)

Aufbau der Query zum Finden beliebiger Ressourcen

```
<xs:element name="ResolverQuery" type="jxta:ResolverQuery"/>
<xs:complexType name="ResolverQuery">
  <xs:all>
    <xs:element ref="jxta:Cred" minOccurs="0"/>
    <xs:element name="SrcPeerID" type="jxta:JXTAID"/>
    <!-- This could be extended with a pattern restriction -->
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="HC" type="xs:unsignedInt"/>
    <xs:element name="Query" type="xs:anyType"/>
  </xs:all>
</xs:complexType>
```

Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE jxta:ResolverQuery>
<jxta:ResolverQuery xmlns:jxta="http://jxta.org">
  <HandlerName>
    urn:jxta:uuid-yyyyyyyyyyyyyy05
  </HandlerName>
  <QueryID>
    urn:jxta:uuid-xxxxxxxxxxxxxxx
  </QueryID>
  <HC>
    5
  </HC>
  <SrcPeerID>
    urn:jxta:uuid-zzzzzzzzzzzzzzz03
  </SrcPeerID>
  <Query>
    ...
  </Query>
</jxta:ResolverQuery>
```

Aufbau der Antwort auf eine ResolverQuery:

```

<xs:element name="ResolverResponse" type="ResolverResponse"/>
<xs:complexType name="ResolverResponse">
  <xs:all>
    <xs:element ref="jxta:Cred" minOccurs="0"/>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="Response" type="xs:anyType"/>
  </xs:all>
</xs:complexType>

```

Endpoint Routing

Endpoint Adressen:

```

http://10.1.1.11:9704/endpoint/resolver
urn:jxta:uuid-pppppppppppppppppppppp04?pipeService
urn:jxta:jxta-NetGroup?relay/uuid-iiiiiiiiiiiiiiiiiii03

```

Route Advertisement:

```

<xs:element name="APA" type="jxta:APA"/>
<xs:complexType name="jxta:APA">
  <xs:sequence>
    <xs:element name="EA" type="jxta:JXTAID"
      minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="RA" type="jxta:RA"/>
<xs:complexType name="jxta:RA">
  <xs:sequence>
    <xs:element name="DstPID" type="xs:anyURI"/>
    <xs:element ref="jxta:RA"/>
    <xs:element name="Hops" minOccurs="0">
      <xs:sequence>
        <xs:element ref="jxta:APA"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

Endpoint Routing Query

```

<xs:element name="ERQ" type="jxta:ERQ"/>
<xs:complexType name="jxta:ERQ">
  <xs:sequence>
    <xs:element name="Dst" type="jxta:JXTAID"/>
    <xs:element name="Src">
      <xs:element ref="jxta:RA"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

Endpoint Route Response

```

<xs:element name="ERR" type="jxta:ERR"/>
<xs:complexType name="jxta:ERR">
  <xs:sequence>

```

```
<xs:element name="Dst">
  <xs:element ref="jxta:RA"/>
</xs:element>
<xs:element name="Src">
  <xs:element ref="jxta:RA"/>
</xs:element>
</xs:sequence>
</xs:complexType>
```

Endpoint Router Transport Protocol

Dient dem Herstellen von virtuellen Transport-Verbindungen zwischen Peers, die nicht direkt eine TCP/IP oder HTTP Verbindung aufbauen können.

weitere Bestandteile des JXTA Protokolls

- Messages
- Peer Discovery
- Rendezvous
- Pipe Binding
- Peer Information
- Message Transport:
TCP/IP, HTTP, TLS, Endpoint Router Transport Protocol
- Message Wire Format

Zusammenfassung und Ausblick

- JXTA P2P als Basis für viele Anwendungen
- Sicherheitskonzept durch konsequente Verwendung von PeerGruppen
- im Unterschied zu Jini nicht an Java gebunden
- mit JXTA Shell und Instant JXTA existieren schöne Beispiele
- auch für J2ME und damit für mobile Geräte geeignet

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Sat May 22 12:42:01 CEST 2004

26. P2P Suche

- Einleitung
 - verteilte Hashtabellen
-

26.1. Einleitung

- Napster: (verteilte) zentrale Verzeichnisse
 - Gnutella: Anfrage an möglichst viele Knoten (#nachbarnTTL)
 - JXTA: Anfrage an möglichst viele Knoten
 - Linda, Jini: Tuple-Spaces
 - Chord: verteilte Hashtabellen (DHT)
-

26.2. Verfahren

- Flooding mit Queries
wie Gnutella und JXTA, Queries an alle Nachbarn mit bestimmter TTL
- Expanding Ring
Queries an alle Nachbarn mit wachsender TTL
- Random Walk
Queries nehmen zufällige Routen (Wege), evtl. werden mehrere zufällige Routen gleichzeitig begangen
Optimierung durch periodische Rückfragen beim Absender
- distributed Hashtables
in P2P Systemen bei denen die Knoten Informationen über die Nachbarschaftsbeziehungen pflegen

verteilte Hashtabellen (distributed hash tables DHT)

- Hashwert zu einem Suchbegriff: s
- Hashwert zu Knoten-Id: k
- Entfernung zwischen Knoten und Suchbegriff: $d(s,k)$
- z.B. Anzahl der Bits, die verschieden sind
- der Knoten mit der geringsten Entfernung zum Suchbegriff wird ausgezeichnet als *Anker*
- der Knoten, der ein Objekt zu dem Suchbegriff speichert, ist ein *Target*
- auf allen Knoten zwischen einem Target und einem Anker werden Verweise auf das Target gespeichert
- eine Suchanfrage wird in Richtung auf einen Anker geroutet
- wird ein Verweis angetroffen, ist das Objekt zum Suchbegriff gefunden und der Verweis wird zum Suchenden geroutet
- über den Verweis kann das Objekt vom Target geladen werden
- hat der Anker keinen Verweis, ist das Objekt nicht erreichbar / vorhanden
- in den Routingtabellen müssen Informationen über 'Entfernungen' zu Suchbegriffen vorhanden sein
- Systeme: Chord, Pastry, Tapestry
 n = Anzahl der Knoten, Speicher pro Node $O(\log(n))$, Lookup $O(\log(n))$, Insert $O(\log(n))$, Background Messages $O(1)$, Background Latency $O(1)$

- Andere Systeme: Kelips
n = Anzahl der Knoten, Speicher pro Node $O(\sqrt{n})$, Lookup $O(1)$, Insert $O(1)$, Background Messages $O(\log(n))$, Background Latency $O(\log(n))$

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Sun Jun 20 14:11:42 CEST 2004

27. Hypertext Transfer Protocol (HTTP)

- Einleitung
 - HTTP Konzepte
 - Web-Server
 - Ausblick
-

27.1. Einleitung

- Clients und Server
 - Anfrage (Request) und Antwort (Response)
 - auf TCP/IP aufbauend
 - eine Sitzung pro Anfrage bei HTTP 1.0
 - bei HTTP 1.1 mehrere Anfragen pro Sitzung
 - Zustandsloses Protokoll, Idempotenz
-

27.2. HTTP Konzepte

- HTTP 0.9
- HTTP 1.0 vom Februar 1996
- HTTP 1.1 vom Juni 1999

HTTP spezifiziert die Syntax und Semantik von Anfragen und Antworten.

- Definition von URLs
- Datumsformat (RFC 822, 1123)
- Zeichensätze (US-ASCII)
- Format der Header
- HTTP 1.1: Persistent Connections
- HTTP 1.1: Content Negotiation
- HTTP 1.1: Caching
- HTTP 1.1: Multi-homed Servers

Ablauf einer Anfrage

www.uni-mannheim.de, localhost

Telnet-Verbindung an den Port 80

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
```

Hypertext Transfer Protocol (HTTP)

```
User-Agent: Heinz Kredel  
Host: localhost  
Accept: */*
```

Antwort des Web-Servers

```
HTTP/1.1 200 OK  
Date: Sat, 07 Nov 1998 20:53:21 GMT  
Server: Apache/1.2.4 S.u.S.E./5.1  
Last-Modified: Thu, 21 May 1998 19:19:48 GMT  
ETag: "21047-44c-35647e54"  
Content-Length: 1100  
Accept-Ranges: bytes  
Connection: close  
Content-Type: text/html  
  
<HTML>  
...  
</HTML>  
Connection closed by foreign host.
```

Die Schritte der Anfrage im Detail

1. TCP/IP-Verbindung (z.B. telnet) an einen verabredeten Port (default 80)
erste Zeile: *method*-Kommando mit Parametern

```
GET / HTTP/1.0
```

2. folgende Zeilen: Informationen des Browsers (UA) an den Server, abgeschlossen durch eine Leerzeile.

```
User-Agent: Heinz Kredel  
Host: localhost  
Accept: */*
```

3. Abhängig von *method* folgen weitere Informationen.
bei *POST* z.B. die Daten des Formulars, oder auch eine Datei.

Allgemein:

```
Method-Line  
General-Header(s)  
Request-Header(s)  
Entity-Header(s)  
CRLF  
Entity-Body
```

Die Schritte der Antwort des Servers

1. erste Zeile: Statuszeile über Erfolg oder Fehler

```
HTTP/1.1 200 OK
```

2. folgende Zeilen: Header-Informationen des Servers, abgeschlossen durch eine Leerzeile.

```
Date: Sat, 07 Nov 1998 20:53:21 GMT  
Server: Apache/1.2.4 S.u.S.E./5.1  
Last-Modified: Thu, 21 May 1998 19:19:48 GMT  
ETag: "21047-44c-35647e54"  
Content-Length: 1100  
Accept-Ranges: bytes  
Connection: close  
Content-Type: text/html
```

Hypertext Transfer Protocol (HTTP)

3. Abhängig von *Content-Type* folgen weitere Informationen.
zum Beispiel bei *text/html* eine HTML-Datei, oder bei *image/gif* ein GIF-Bild.

Allgemein:

```
Status-Line  
General-Header(s)  
Response-Header(s)  
Entity-Header(s)  
CRLF  
Entity-Body
```

Nach dem Abbau der Verbindung gibt es auf beiden Seiten keine Informationen mehr über den Partner.

Methoden

- GET
Abholen von Informationen
- HEAD
Abholen der Header-Informationen
- POST
Mitschicken von Informationen an den Server
- PUT, DELETE
fast nie verwendet, nicht für HTTP 1.0 notwendig
- LINK, UNLINK
fast nie verwendet, nicht für HTTP 1.0 notwendig

Statusmeldungen des Servers

- 100-199
zur Information
- 200-299
Client Anfrage erfolgreich
- 300-399
Client Anfrage umgeleitet, eine neue Anfrage ist erforderlich

```
300 Multiple Choices  
302 Moved Temporarily  
304 Not Modified
```

- 400-499
Client Anfrage unvollständig

```
400 Bad Request  
401 Unauthorized  
403 Forbidden  
404 Not Found
```

- 500-599
Server Fehler

```
500 Internal Server Error  
503 Service Unavailable
```

HTTP Header

Allgemeine Header

- Connection:
Close, Keep Alive
- Date:
Zeit und Datum
- MIME-Version:
- Transfer-Encoding:
- diverse Cache Optionen

Client Anfragen

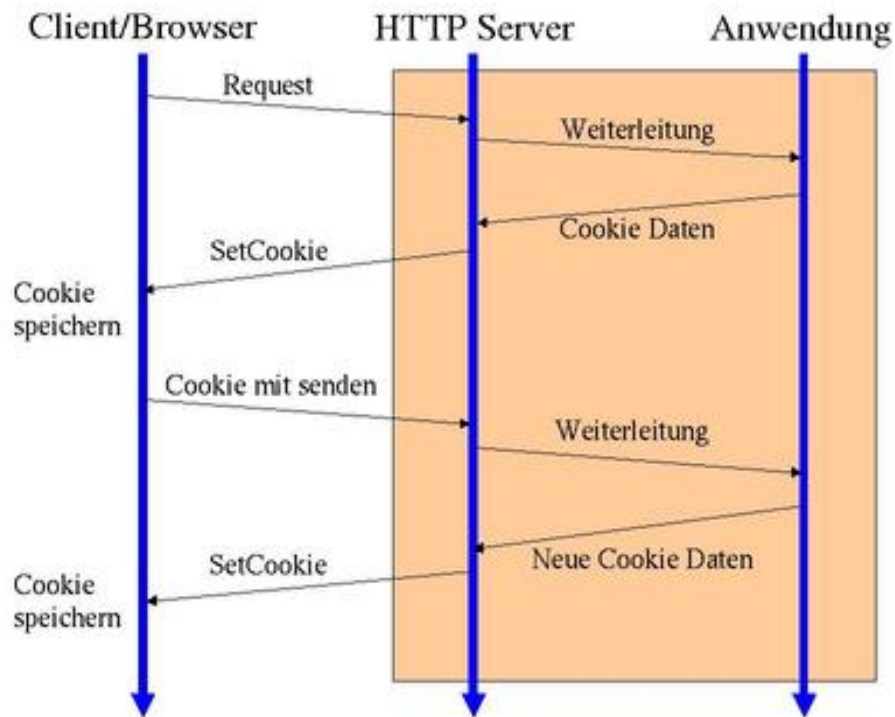
- Accept: type/subtype
MIME-Types die angenommen werden
- Authorization: Basic base64(username:password)
Authorization: Digest MD5(MD5(uname):number:MD5(passwd))
- Cookie: name=wert
- If-Modified-Since: date
- Referer: url
- User-Agent: Zeichenkette
- Host: hostname:port
muß bei HTTP 1.1
- diverse Cache und Match Optionen

Server Antworten

- Retry-After: date|seconds
nochmal versuchen bei 503
- Server: Zeichenkette
- Set-Cookie: name=wert; options
expires=date, domain=domain_name, path=pathname
- WWW-Authenticate: Basic realm="Bezug"
WWW-Authenticate: Digest realm="Bezug" nonce="nummer"

Cookies

Cookies



Inhaltsangaben

- Content-Type: mimetype
- Content-Length: bytes
- Content-Encoding: scheme
x-gzip, x-www-form-urlencoded
- Content-Transfer-Encoding: scheme
8bit, base64, quoted-printable
- Content-Language: sprache
- Expires: date
- Last-Modified: date
- Location: url

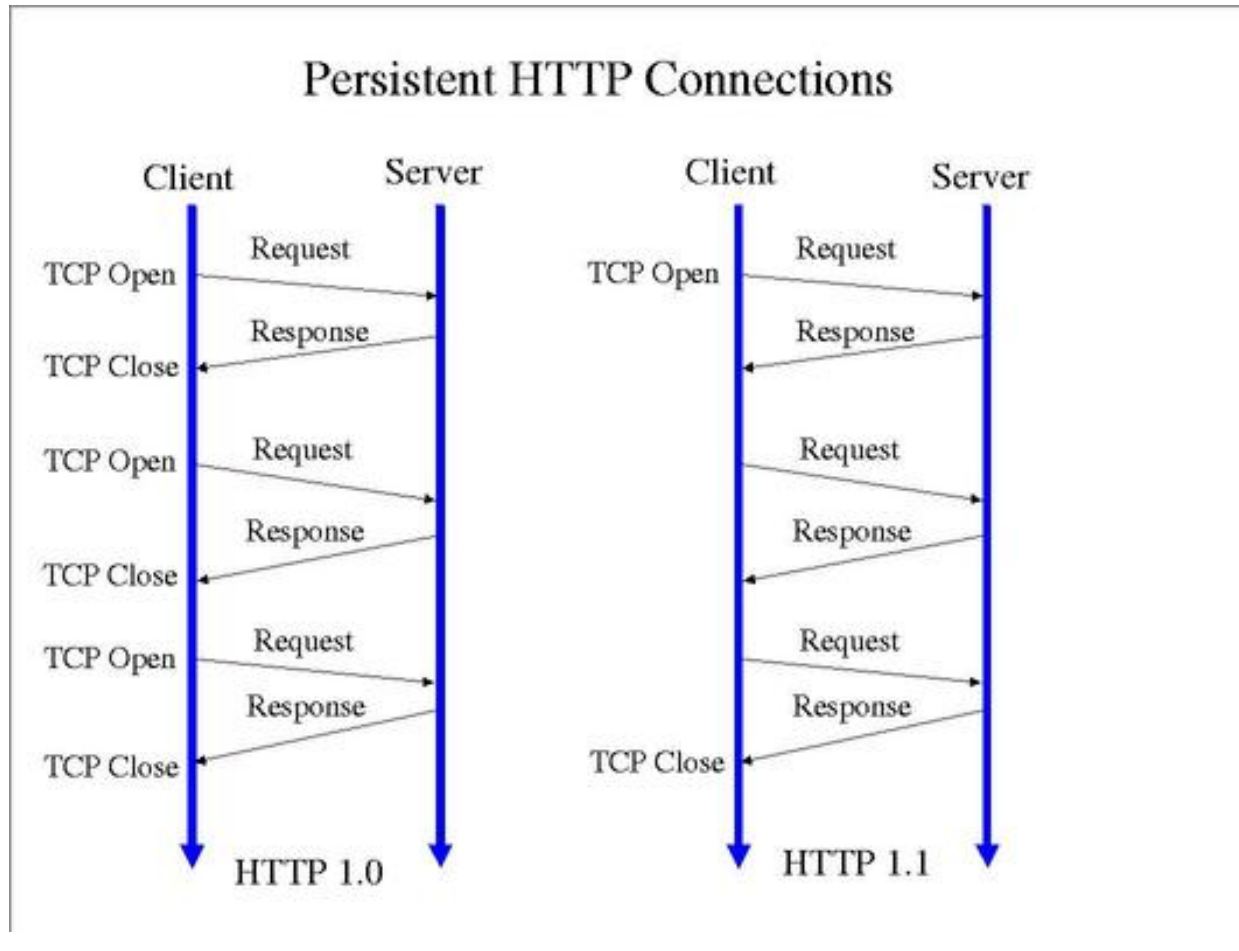
MIME-Types

Multipurpose Internet Mail Extensions

Typ/Subtyp	Datei-Erweiterung	Datei-Typ
application/pdf	pdf	Portable Document Format von Adobe
application/ps	eps, ps	PostScript

application/x-tar	tar	"Tape-Archive"
audio/basic	au, snd	Audioformat
image/gif	gif	Graphikformat
image/tiff	tiff, tif	Graphikformat
text/html	html, htm	HTML Datei
text/plain	txt	reine ASCII Datei

Verbesserungen in HTTP 1.1



27.3. Web-Server

- CERN Server
- NCSA Server
- Roxen
- Netscape Server
- Microsoft Server

- IBM Server
- MOWS, Web-Server in Java
- Jigsaw, Web-Server in Java vom W3C
- Xitami
- Apache

Arbeitsweise

- wartet auf TCP/IP-Verbindung
- Umsetzen des URLs
`http://host/path/file.html`
bzw. `GET /path/file.html HTTP/1.1`
- auf Datei-System: Document-Root
`/disk1/www/htdocs/path/file.html`
- ausliefern der Datei

Apache Entwicklung

- Anfang als Patch (Verbesserungen) zu NCSA Server "a patchy server"
- Februar 1995 basierend auf NCSA 1.3
- Dezember 1995, Apache 1.0
- 1999 Gründung der Apache Software Foundation
- zur Zeit Version 1.3.27
- neueste Version 2.0.44
- Multithreading wo möglich
- bessere Unterstützung von nicht-Unix Systemen (Windows, OS/2, BeOS)
- neues API (Programmierschnittstellen)
- Unterstützung für IPv6
- Filter Module
- wird als Quellcode und Binär angeboten
- frei verfügbar
- über 50 % Marktanteil
- Unterstützt alle wichtigen Dinge: CGI, URL Rewriting, Perl, PHP, Security mit OpenSSL, etc.
- stark Modularisiert

(Apache) Konfiguration

Ältere Versionen (wie NCSA):

- `httpd.conf`
Server Grundkonfiguration
- `srm.conf`
Konfiguration der Dateien und anderen Ressourcen
- `access.conf`
Konfiguration der Zugriffsrechte
- `mime.types`
Zuordnung von MIME-Types zu Datei-Extensions

Aktuelle Versionen:

- Globale Definitionen
- Definitionen für den Haupt-Server
- Definitionen für optionale virtuelle Server
- mime.types
Zuordnung von MIME-Types zu Datei-Extensions

Beispiel: [httpd.conf](#) und [mime.types](#)

Sicherheit

- httpd.conf (früher in access.conf)

```
<Directory /usr/local/httpd/htdocs/secure>
Options FollowSymLinks
AllowOverride AuthConfig
order allow,deny
allow from all
</Directory>
```

- /usr/local/httpd/htdocs/secure/.htaccess

```
AuthUserFile /usr/www/etc/passwd
AuthGroupFile /usr/www/etc/groups
AuthName MyIntern
AuthType Basic

require valid-user
```

Bessere Authentifizierung mit 'AuthType Digest'. Bei Bedarf auf bestimmte Methoden beschränken:

```
<Limit GET>
require valid-user
</Limit>
```

- passwd

```
name1: xyz
name2: xyz2
```

- group

```
members: name1, name2
```

Ablauf:

1. Zugriff auf geschützte Seite
2. Server schickt Fehlermeldung 401 Unauthorized und verlangt Authentifizierung
3. Browser fragt mit Dialogbox nach den Informationen, z.B. Benutzer/Passwort
4. Browser wiederholt die Anfrage nach der geschützten Seite, diesmal aber mit Authentifizierung
5. falls OK, schickt der Server die Seite

Installation

- erhältlich für fast alle Betriebssysteme
- Compilierung durch Make-Files einfach

- Konfiguration mit Textdateien und Direktiven
- Start, Stop und Restart
- viele Logfiles
- Kern und Module
- Virtuelle Hosts
- Handler

Log-Files

httpd.error_log:

```
[Mon Dec 20 08:29:18 1999] [notice] Apache/1.3.6 (Unix) (SuSE/Linux) configured -- resuming normal operation
[Mon Dec 20 08:29:18 1999] [notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
[Tue Dec 21 00:24:23 1999] [notice] caught SIGTERM, shutting down
...
[Tue Dec 21 11:38:59 1999] [notice] Apache/1.3.6 (Unix) (SuSE/Linux) configured -- resuming normal operation
[Tue Dec 21 11:38:59 1999] [notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
```

httpd.access_log:

```
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET / HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET /gif/gl.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET /gif/apache_logo.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET /gif/awlogo.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET /gif/suse_150.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:20:39:33 +0100] "GET /gif/apache_pb.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:23:03:29 +0100] "GET /manual/ HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:23:03:29 +0100] "GET /manual/images/sub.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:23:03:29 +0100] "GET /manual/images/index.gif HTTP/1.0" 304 -
127.0.0.1 - - [21/Dec/1999:23:03:45 +0100] "GET /manual/install.html HTTP/1.0" 200 10455
127.0.0.1 - - [21/Dec/1999:23:04:13 +0100] "GET /manual/new_features_1_3.html HTTP/1.0" 200 33665
```

Auswertung der Log-Files

z.B. Webalizer

27.4. Ausblick

- HTTP-NG
 - Distributed Object System
 - Protocol Extension Protocol (PEP)
 - Performance Benchmarks
 - Common Gateway Interface (CGI)
 - Secure Socket Layer (SSL)
 - Server Side Includes (SSI)
 - Apache Module
 - Jigsaw, Web-Server in Java vom W3C
 - Java-Servlets
-

Last modified: Sat May 22 12:36:52 CEST 2004

28. Interaktion und CGI

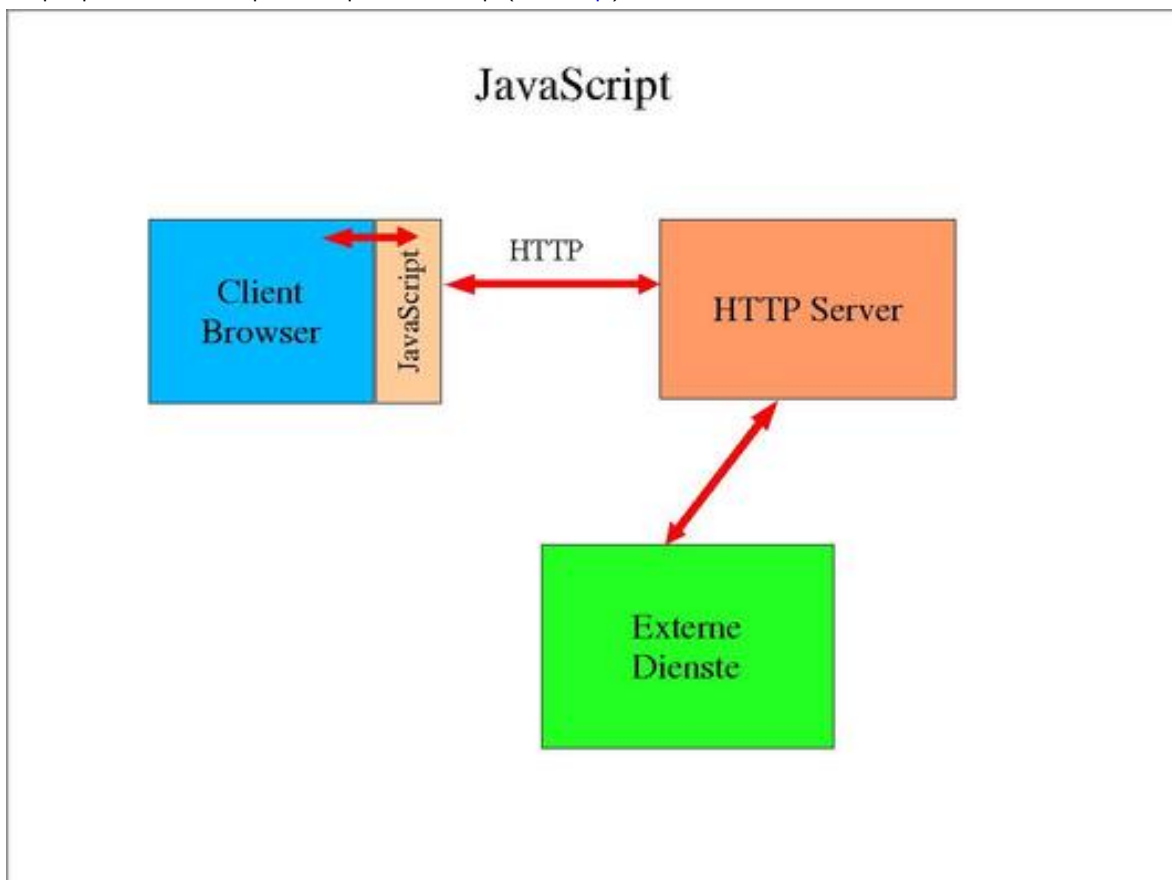
- Interaktion mit externen Diensten
 - Konzepte des Common Gateway Interface (CGI)
-

28.1. Interaktion mit externen Diensten

- HTML bietet "nur" statische Seiten
- Browser Erweiterungen
JavaScript, (Netscape) Plugins, Java Applets
- Server Erweiterungen
CGI Common Gateway Interface,
PHP, Java Servlets,
HTTPD APIs, Application Programming Interfaces

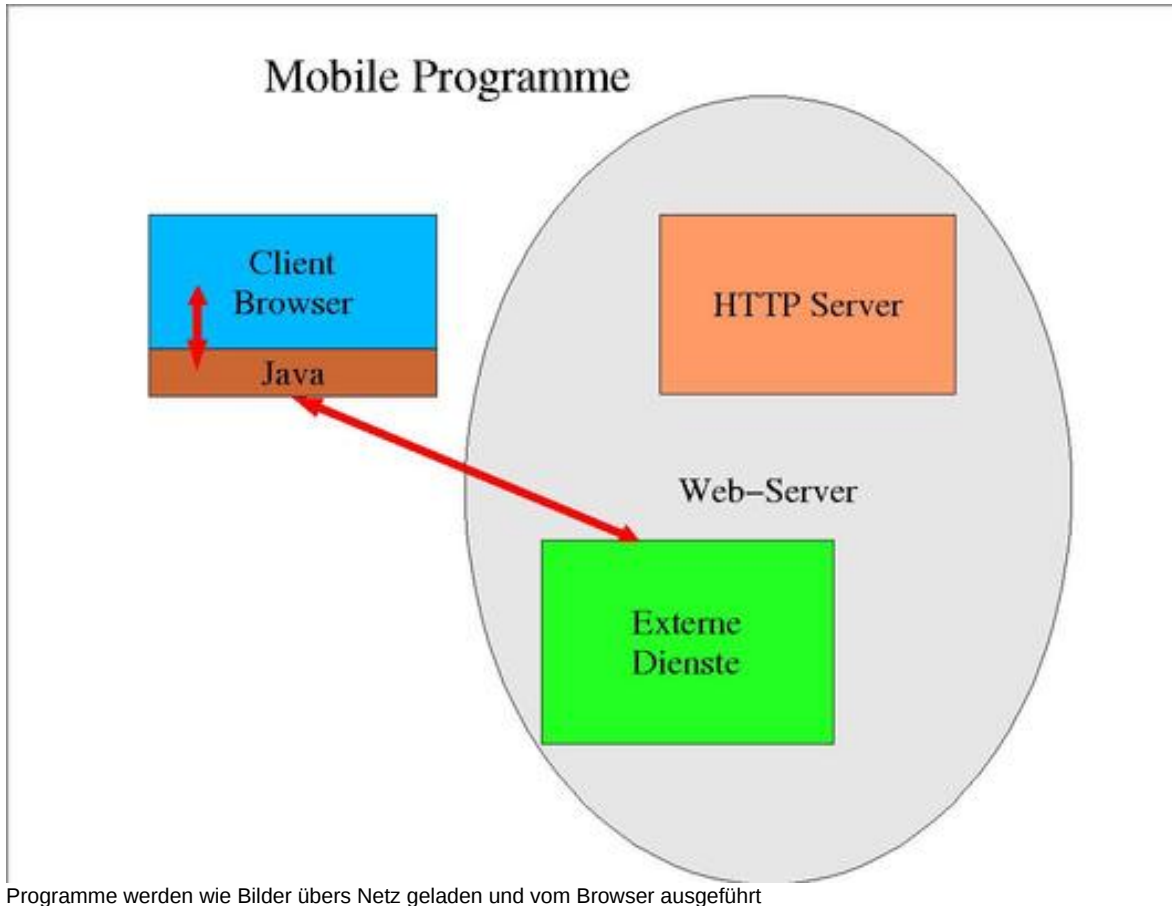
Browser Erweiterungen

- Script-Sprachen: JavaScript, VBScript, ECMA-Script ([JavaScript](#))

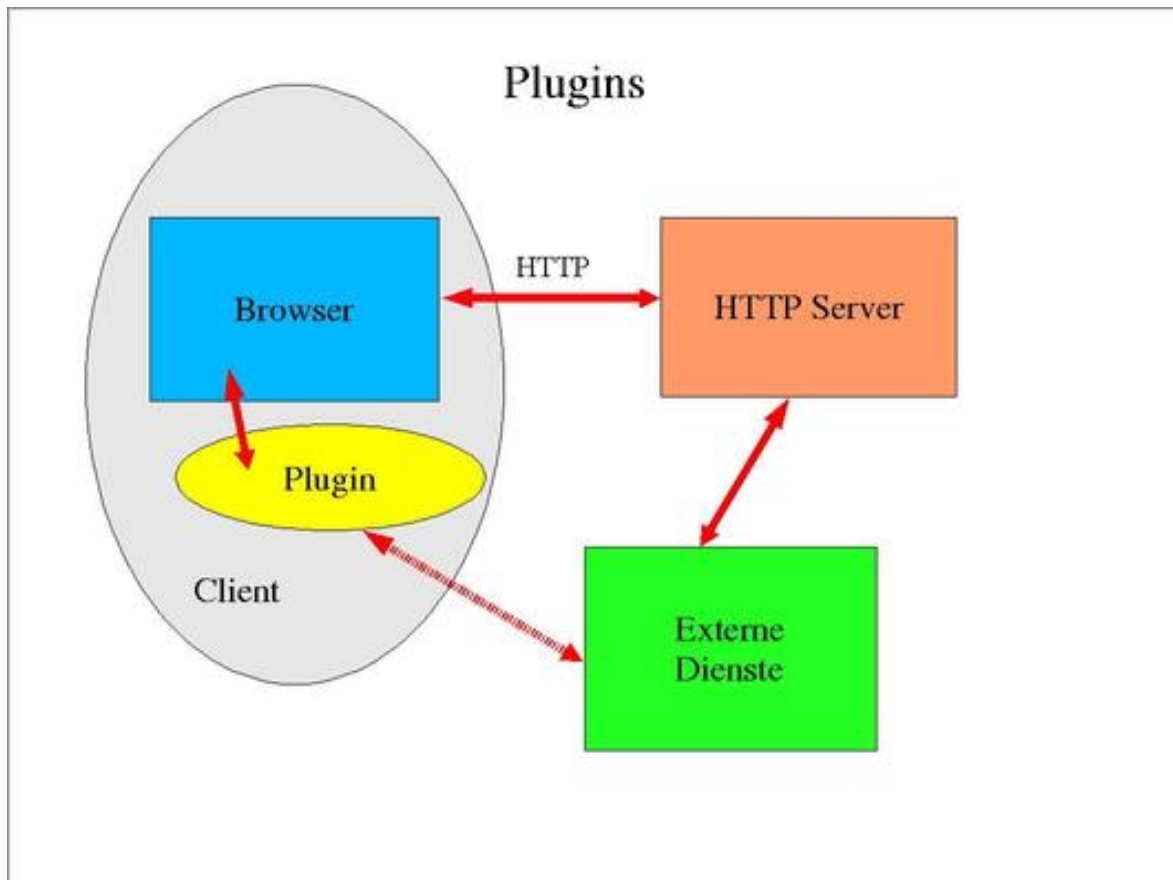


Programmteile werden in HTML Seiten eingebettet

- Vorteile:
Einfache Erweiterung des Browsers
z.B. Feldprüfungen
weit verbreitet, System unabhängig
nicht das gesamte Interface muß programmiert werden
- Nachteile:
eingeschränkter Funktionsumfang
- Java Applets ([Java](#))



- Vorteile:
Effizient
Bessere Oberfläche (GUI) realisierbar
Status-Information werden gewartet
- Nachteile:
das gesamte Interface muß programmiert werden
- Beispiele [Financial Portfolio](#), [Applets](#)
- (Netscape) Plugins

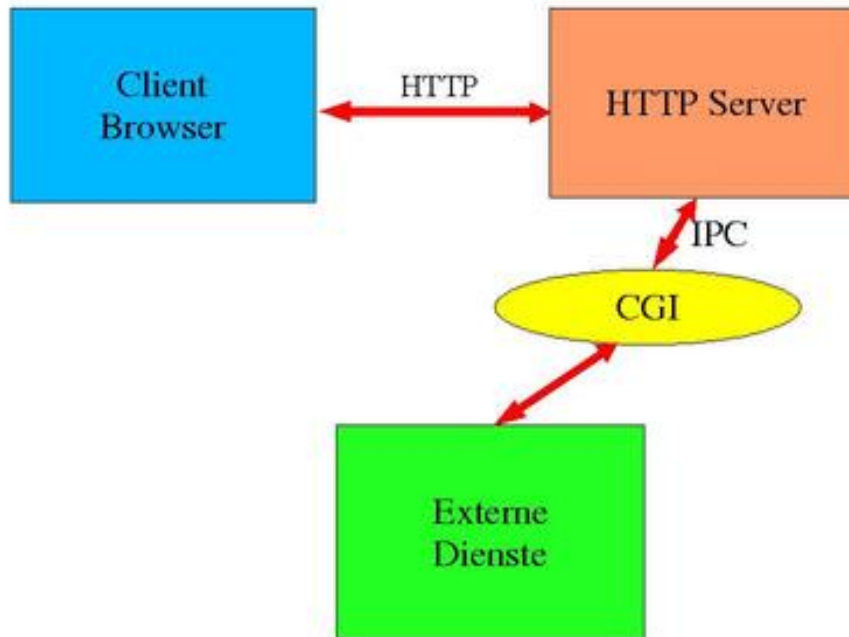


Programme müssen lokal installiert werden und werden vom Browser wie externe Viewer geladen und ausgeführt

- Vorteile:
Mächtige Erweiterung des Browsers möglich
z.B. Virtual Reality Extensions
- Nachteile:
nur Netscape
abhängig vom Betriebssystem
muss vollständig neu programmiert werden

Common Gateway Interface (CGI)

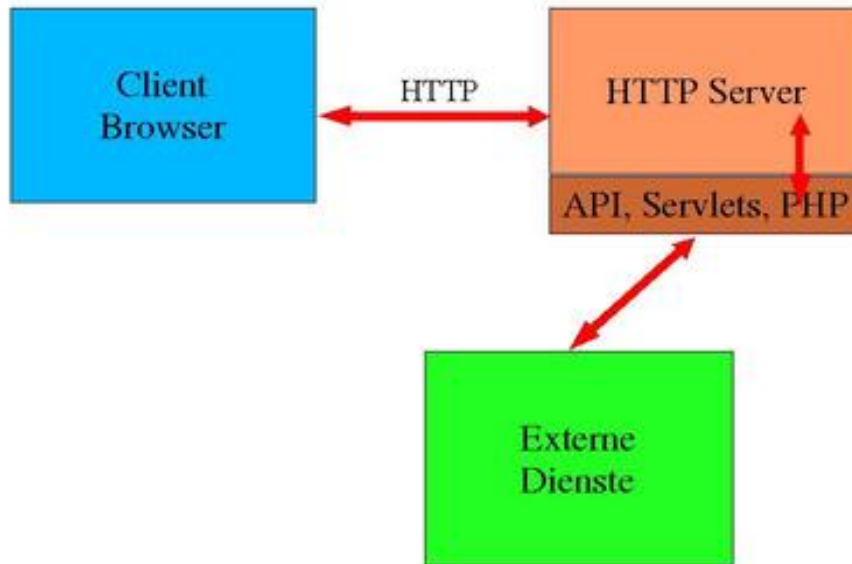
Common Gateway Interface



- Gateway Programme:
Skripten, Umgebungsvariablen
- Vorteile:
schnell zu Erstellen
einfacher Zugang, alles machbar
unabhängig vom WWW Server
- Nachteile:
Belastet WWW Server
Verwaltungsaufwand für Status-Informationen
- Anwendungen:
[CONSULT-Web](#), [FZ Data-Warehouse](#), [OSIS Online Steel Information System](#), [Lehmanns Online Bookshop](#).

Integration in Server

Server Erweiterungen



- Application Programming Interface (API)
[NSAPI](#) von Netscape, [ISAPI](#) von Process Software, [Apache-API](#) von Apache
- Server Side Includes (SSI)
[PHP](#), Active Server Pages (ASP) von MS, Java Server Pages (JSP), LiveWire von Netscape
- Java Servlets
- Vorteile:
Effizient
Status-Information werden verwaltet
- Nachteil:
teilweise Abhängigkeit vom Server

28.2. CGI Konzepte

Einleitung

Was ist das CGI ?

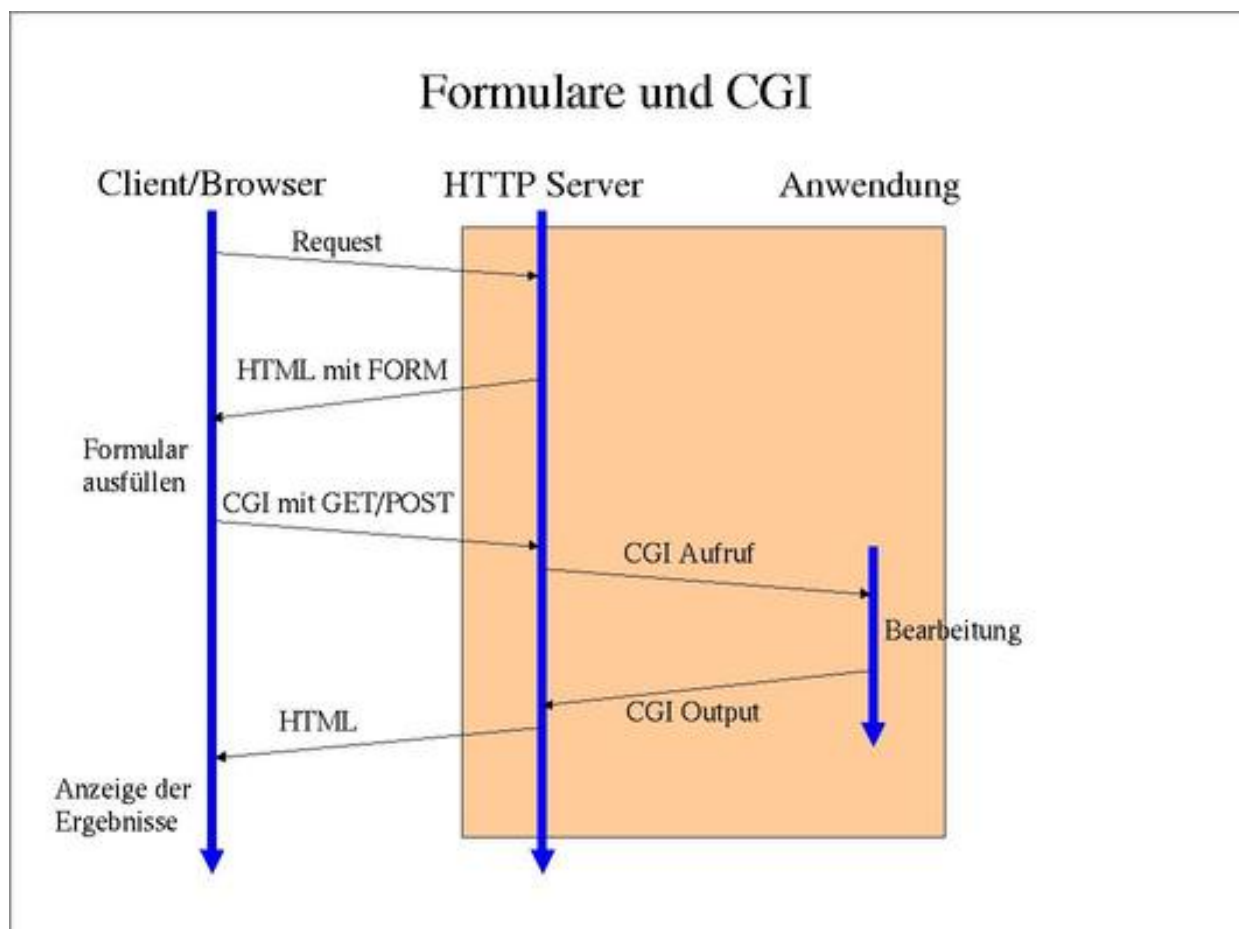
Common Gateway Interface

- ermöglicht beliebige vorbereitete Programme auszuführen.

- Eingabeinformationen vom WWW-Client
- liefern HTML Daten an den WWW-Client zurück
- *spezifiziert* erforderliche Schnittstellen

Wozu kann das CGI verwendet werden ?

- Bereitstellung von HTML Informationen *on the fly*
- *interaktive Antworten*
- Konvertierung von Handbuchseiten in HTML versenden,
- Verbindung zu WAIS,archie, SQL, Datenbank;
Stellen einer Anfrage an die Datenbank
Aufbereiten der Antwort in HTML
- Interaktion mit Benutzern des WWW-Servers, z.B. Warenauswahl und Bestellung.
- non-html Dokumente: *Volltext Dokumenten-Systeme*.



Wie sehen CGI Programme aus ?

- *selbständig* vom Betriebssystem ausgeführte Programme
- in jeder gängigen Programmiersprache
- Bedingungen:

1. Umgebungsvariablen (environment variablen)
2. Standart-Eingabe (stdin)
3. Standart-Ausgabe (stdout)

Verbreitete Programmiersprachen

- PERL (Practical Evaluation and Reporting Language),
- diverse Unix Shells: sh, csh, tcsh, bash, ksh, zsh
- REXX (REstructured eXtended eXecutor),
- Python,
- TCL (Tool Command Language),
- C oder C++.

Interpretierte Sprachen bevorzugt.

Wer hat CGI entwickelt ?

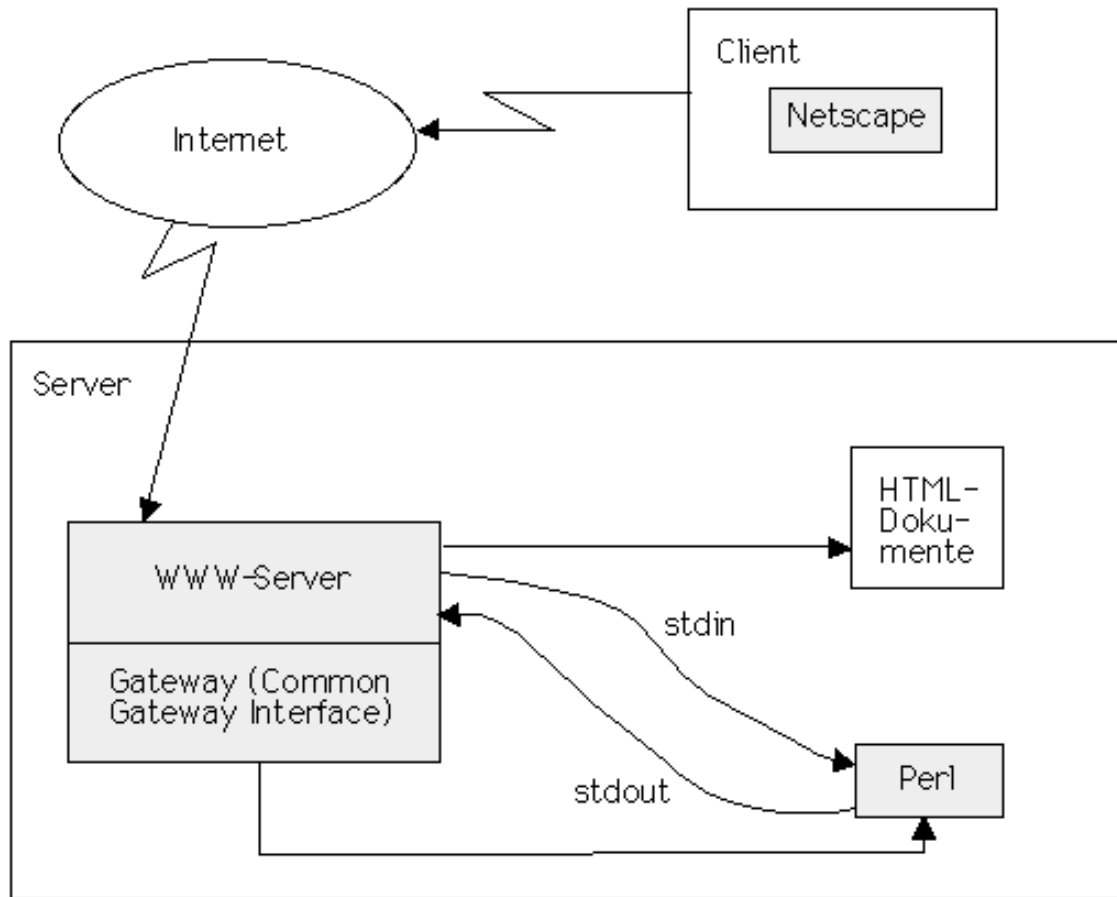
Autoren der ersten WWW-Server

- Tony Sanders,
- Ari Luotonen,
- George Phillips,
- John Franks.

Aktuelle Version CGI/1.1.

Benutzung

Wie werden die Informationen zwischen dem WWW-Client, WWW-Server und dem CGI Program ausgetauscht ?



Auswahl des CGI Programms in einem URL.

`http://server-ip/path/cgi-prog/path-info?query-string`

- Zugriffsmethode (http:),
- Servername bzw. IP-Adresse (`//server-ip`)
- Pfad zur Resource (path).
- Beginnt meist mit `cgi-bin`.
- Konfiguration des WWW-Servers (`httpd.conf`)
- Name des CGI-Programms (`cgi-prog`)
- `.pl` bedeutet ein Perl Program

Hinweis: keine Kommandozeilenparameter an das CGI Program

D.h. `cgi-prog arg1 arg2` geht nicht

Wie wird Information vom Server empfangen ?

Empfang auf 3 Arten:

- QUERY_STRING Umgebungsvariable,
- PATH_INFO Umgebungsvariable,
- direkt über Stdin.

Query-String Methode

`http://server-ip/path/cgi-prog/path-info?query-string`

`query-string` in Umgebungsvariable QUERY_STRING

- URL Kodierungsschema
- Leerzeichen (Blanks) in +
- spezielle Zeichen in hexadezimaler Form %xx
- Dekodierung erforderlich
- Angabe von Hand oder
- vom WWW-Client generiert
- z.B. von <isindex> oder <form>

Path-Info Methode

`http://server-ip/path/cgi-prog/path-info?query-string`

`/path-info` in Umgebungsvariable PATH_INFO

- *nicht* im URL Schema kodiert
- Angabe nur explizit
- z.B. Grundinformationen wie die aktuelle Sprache
- `/cgi-prog/language=english?query-string`.

Stdin-Datei Methode

Wird per Pipe gesendet und von Stdin gelesen

- Kodierung wie im QUERY_STRING
- Dekodierung erforderlich
- Auswahl mit <form> Element und
- Attribut `method="POST"`
- Länge in der Umgebungsvariablen CONTENT_LENGTH
- End-of-File für Stdin ist nicht definiert

Wie wird Information zum Server gesendet ?

Rückgabe durch Stdout "Datei" im richtigen Format: HTML

- Header: 2 Zeilen ASCII Text
- 1. Zeile MIME-Type oder Location
- 2. Zeile leer

MIME Inhaltstyp

Content-Type: m-type/m-stype

Beispiele

- text/html für HTML Text
- text/plain für ASCII Text
- auch für Graphiken, Sound und Videos

Ortsangabe

Location: ftp://host/dir/dateix.txt

Der Client erzeugt eine FTP-Verbindung

Wie wird Information vom Client aufbereitet ?

Formulare: (X)HTML <form> Element

Attribut mit zwei Werten

- method="GET", Variable QUERY_STRING
- method="POST", "Datei" Stdin

Daten des Formulars

- name Attribut: Namen für Eingabefelder z.B. bei <input type="text" name="x" >
- Inhalt der Eingabefelder wird mit Namen gesendet
- Folge von name=content Paaren
- durch & getrennt
- wird automatisch kodiert
- im CGI Program dekodieren und verwenden

Grundaufbau CGI Program

1. Feststellen der Übertragungsmethode
Ansehen der Umgebungsvariablen REQUEST_METHOD.
Mögliche Werte: GET, POST, HEAD
2. Lesen von CONTENT_LENGTH Bytes von QUERY_STRING oder Stdin.
3. URL Dekodierung der Zeichenkette.
4. Aufspaltung der Zeichenkette entlang &.
5. Aufspaltung der Paare name=content entlang =.
6. Verwenden der so aufbereiteten Informationen.

Spezifikation

CGI 1.0 oder 1.1:

- Umgebungsvariablen (environment variables),
- Standard-Eingabe (stdin) sowie
- Standard-Ausgabe (stdout),
- die Kommandozeilenparameter (command line).

Alle Spezifikationen der Version 1.1 sollen auch von zukünftigen Erweiterungen erfüllt werden.

28.3. Zusammenfassung und Ausblick

- Interaktion erfordert Browser oder/und Server Erweiterungen
 - auf Server Seite gibt es viele flexibel einsetzbare Instrumente
 - auf der Browser Seite ist man durch die Möglichkeiten des jeweiligen marktbeherrschenden Browsers eingeschränkt
 - CGI kann zusammen mit vielen Programmiersprachen eingesetzt werden
 - CGI ist nach wie vor eine wichtige Web-Technik
-

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:37:04 CEST 2004

29. Perl und CGI

- Einleitung
 - Sprachkonstrukte
 - Module
 - CGI Anwendung
 - CGI.pm Modul
-

29.1. Einleitung

- eine interpretierte Sprache,
- für Erschließung, Aufbereitung und Neuformatierung beliebiger Textdateien
- kann C, sed, awk und sh ersetzen
- überwindet alle Beschränkungen dieser Sprachen

Perl von Larry Wall entwickelt, Version 5.0.

Auch *Pathologically Eclectic Rubbish Lister*

Verwendung

Perl Dateien haben Endung .pl.

Ausführung mit

```
perl tuwas.pl
```

Unter UNIX andere Variante:

```
#!/usr/bin/perl
```

```
# Unix Kommentar,  
! der Name eines Kommandointerpreters folgt  
/usr/bin/ ist eine Pfadangabe,
```

Kurzes Beispiel:

Frage nach dem Namen und Ausgabe von "Hallo, ... ! "

```
(1) #!perl  
(2) print "Wie ist dein Name ? ";  
(3) $name = <STDIN>;  
(4) chop($name);  
(5) print "Hallo, $name !\n";
```

29.2. Sprachkonstrukte

der *Kontext* in dem ein Literal oder eine Variable verwendet wird, wird durch die verwendeten Operatoren erzwungen.

Beispiel

"33" + "44" ergibt Zahl 77

33 . 44 ergibt Zeichenkette "3344"

Zeichenketten

- 'abc' wie q/abc/, keine Interpolation
- "abc" wie qq/abc/, mit Interpolation
- `COMMAND` wie qx/COMMAND/, Ausführung eines externen Kommandos

Variablen

Beispiel: Arrays, Felder

```
@fred = (11,22,33);  
@barney = @fred;
```

auch mit dem qw/.../ Operator

```
@fred = qw/11 22 33/;
```

Beispiel: assoziatives Array

```
%fred = (1,22,'h',33,"\t",44);  
%barney = %fred;
```

auch mit dem => Operator

```
%fred = (1 => 22, 'h' => 33, "\t" => 44);
```

Beispiel: Skalare

```
print $fred[2] + $fred{'h'};
```

ergibt 33+33 = 66

Skalar-Kontext

```
print @fred * %fred;
```

ergibt 3*3 = 9

Kontrollstrukturen

Statements,

z.B. Zuweisungen VAR = EXPR;

Folgen von Statements { ... },

Statements immer mit Semikolon abgeschlossen

Kontrollstatements

- sub name BLOCK;
Funktionen werden mit dem Schlüsselwort sub definiert. Zum Aufruf von Funktionen wird &name verwendet.

```
sub unter {
    local($a,$b) = @_;
    return "Das Ergebnis ist " . ( $a*$b ) . " wie gewünscht";
}

print "\n" . &unter(6,7) . ".\n\n";

Das Ergebnis ist 42 wie gewünscht.
```

- `if (EXPR) BLOCK [ elsif (EXPR) BLOCK ... [ else BLOCK ] ];`
Im Gegensatz zu C müssen nach `if (EXPR)` immer geschweifte Klammern `{ }` stehen.
- `unless (EXPR) BLOCK [ else BLOCK ];`
Wie `if (NOT EXPR)` .
- `while (EXPR) BLOCK;`
- `for (EXPR; EXPR; EXPR) BLOCK;`
Die For-Schleife ist wie in C.
- `foreach VAR (ARRAY) BLOCK;`
Beispiel: `foreach $a (@fred) { print "$a\n"; }`
- `EXPR || die "Reason";`
Falls die Auswertung von `EXPR` fehlschlägt wird das Perl Program abgebrochen. `Reason` wird auf `STDERR` ausgegeben.

Nach der Auswertung der Ausdrücke `EXPR` bedeuten die leere Zeichenkette `""` , `"0"` und `0` **false**; alle anderen Werte bedeuten **true** (z.B. auch `"00"`).

Match-Operatoren und Variablen Substitutionen

- `VAR =~ m/reg-expr/;`
Sucht nach dem regulären Ausdruck `reg-expr` in der Variablen `VAR`.
- `VAR =~ s/old/new/;`
Substituiert den Ausdruck `old` durch den Ausdruck `new` in der Variablen `VAR`.
- `VAR =~ tr/a-z/A-Z/;`
Ersetzt die entsprechenden Zeichen in der Variablen `VAR`, d.h. `a` wird durch `A` ersetzt usw.

Objekte

- `OBJREF -> METHOD(PARAMETERS);`
Aufruf der Methode `METHOD` mit den Parametern `PARAMETERS` des Objekts `OBJREF`.
- `VAR = OBJECT -> new(PARAMETERS);`
Erzeugen einer neuen Objektreferenz in `VAR` des Objekts `OBJECT`.

Ein- und Ausgabe

Dateien, Pipes und Filehandles.

- `open( FILEHANDLE, "name" );`
`open( FILEHANDLE, "<name" );`
Öffnet die Datei `name` zum lesen.
- `open( FILEHANDLE, ">name" );`
(Erzeugt und) öffnet die Datei `name` zum (über)schreiben.
- `open( FILEHANDLE, ">>name" );`
Öffnet die Datei `name` zum schreiben, der alte Inhalt bleibt erhalten.
- `open( FILEHANDLE, "+<name" );`
Öffnet die Datei `name` zum lesen und schreiben.
- `open( FILEHANDLE, "+>name" );`
(Erzeugt und) öffnet die Datei `name` zum lesen und schreiben.

- `open( FILEHANDLE, "+>>name" );`
(Erzeugt und) öffnet die Datei `name` zum lesen und schreiben, der alte Inhalt bleibt erhalten.
- `open( FILEHANDLE, "|name" );`
Startet das Programm `name` und öffnet eine Pipe zum schreiben auf `STDIN` dieses Programms.
- `open( FILEHANDLE, "name|" );`
Startet das Programm `name` und öffnet eine Pipe zum lesen von `STDOUT` dieses Programms.
- `close( FILEHANDLE );`
- `<FILEHANDLE>`
Liefert die nächste Zeile der Datei.
- `print( FILEHANDLE, EXPR );`
Schreibt auf die Datei `FILEHANDLE`, ohne `FILEHANDLE` wird auf `STDOUT` geschrieben.
- `read( FILEHANDLE, VAR, LENGTH );`
Liest `LENGTH` Bytes von Datei `FILEHANDLE` in die Variable `VAR`.

Print-Formatierung

Zusammen mit HTML überflüssig.

```
format FILEHANDLE =
fieldline_1
valueline_1

fieldline_n
valueline_n
.
```

Definiert ein Printformat für die Datei `FILEHANDLE`. `fieldline_i` definiert das Aussehen einer Zeile und `valueline_i` listet alle Werte und Variablen, die ausgegeben werden sollen.

- Zum Beispiel durch das Format

```
format STDOUT =
@#### @<<<<< @|||| @>>>>>
$a, $b, $b, $b
.
```

wird die Ausgabe der Werte der Variablen `$a` und `$b` auf `STDOUT` definiert. Dabei wird `$a` als Zahl in einem Feld der Länge 5 formatiert, das erste `$b` wird als Zeichenkette linksbündig in einem Feld der Länge 6 formatiert, das zweite `$b` wird als Zeichenkette zentriert in einem Feld der Länge 6 formatiert, schliesslich wird das letzte `$b` als Zeichenkette rechtsbündig in einem Feld der Länge 6 formatiert,

- `format FILEHANDLE_TOP =`
Definiert ein Format für den Kopf einer Ausgabeseite.
- `write FILEHANDLE;`
Damit wird ein Datensatz entsprechend dem Format mit den aktuellen Werten der Variablen geschrieben.

Perl Funktionen reichen von arithmetischen Funktionen und Funktionen für Zeichenketten bis zu Datenbank, Netzwerk und Interprozesskommunikations Funktionen.

29.3. Standard Module

Verwendung / Import von Modulen durch
`use MODULE CONFIGLIST;`

Module sind meist Objektorientiert implementiert
`VAR = MODULE -> new();`

VAR -> method(param);

- CGI: Web Server Common Gateway Interface.
- CPAN: Interface to Comprehensive Perl Archive Network.
- Config: Interface zur Perl Konfiguration.
- ExtUtils: Installation, Makefiles, C Programmierung.
- File: Dateihandhabung
- GetOpt: Auswerten von Kommandozeilen Parametern
- IO: Ein-/Ausgabe auf Dateien und Sockets.
- IPC: Inter Process Communication.
- Math: Mathematische Funktionen.
- Net: Netzwerk Hilfsmittel.
- Term: Terminal IO.
- Text: Textprocessing Hilfsmittel.
- Time: Hilfsmittel für Zeitverwaltung.
- User: Hilfsmittel für Benutzerverwaltung.

29.4. CGI Anwendung

Perl Skript, das mit CGI zusammenarbeitet. Es dekodiert die von CGI empfangenen Variablen und Werte, generiert eine HTML Seite mit diesen Informationen und schickt sie an den Absender zurück.

```
#!/usr/bin/perl
# test perl program to be used for parsing CGI methods
# send anything to this script either via GET or POST methods

&InsertHeader("CGI generated text");
&Parse;
&InsertTrailer;

# subroutines
sub Parse {
    local(@pairs,$pair,$val,$pos, $i);

    if ($ENV{'REQUEST_METHOD'} eq "GET") {
        $in = $ENV{'QUERY_STRING'};
        print "Submitted via GET<P>\n";
    }
    elsif ($ENV{'REQUEST_METHOD'} eq "POST") {
        read(STDIN, $in, $ENV{'CONTENT_LENGTH'}+1);
        print "Submitted via POST<P>\n";
    }

    $i = 0;
    @pairs = split(/&/, $in);

    foreach $pair (@pairs) {
# do the special character conversion
        $pair =~ tr/+// /;
        $pair =~ s/%(..)/pack("c",hex($1))/ge;

        $pos = index($pair,"=");
        $name = substr($pair,0,$pos);
        $val = substr($pair,$pos+1);

        $i++;
        $entry{$name} = $val;
    }
}
```

```

        print "$i: entry\\\\"$name\\"\\} = $entry{$name}<BR>\n";
    }
    return 1;
}

sub InsertHeader {
    local ($htmltitle) = @_;
    print "Content-type: text/html\n\n";
    print "<HTML>\n<HEAD>\n<TITLE> ";
    print "$htmltitle </TITLE>\n</HEAD>\n";
    print "<BODY>\n";

    return 1;
}

sub InsertTrailer {
    print "</BODY>\n</HTML>\n";

    return 1;
}

```

&InsertHeader; bezeichnet eine Funktion, die einen korrekten HTML Header erzeugt. &InsertTrailer; erzeugt den letzten Teil der HTML Seite. &Parse; zerlegt die CGI-Parameter wie im Abschnitt zu CGI besprochen.

Minimales CGI/Perl

Beispiele mit dem Script [ex1.cgi](#)

Zum Ausdrucken der Umgebungsvariablen klicken Sie [hier](#).

Zum Ausdrucken der Umgebungsvariablen mit Querystring klicken Sie [hier](#).

Das nächste Script benutzt ein Formular mit "Radio Buttons".

Mein Name: **Textfield:**

Mein Status: Student Mitarbeiter Professor

Ein neues Formular mit "Check Boxen", "PopUp Selektoren" und "Textbereichen".

Mein Name: **Textfield:**

Mein Status: Student Mitarbeiter Professor

Meine Hobbies: Fußball Reisen Lesen Computer

Perl mit Dekodierung

Beispiele mit dem Script [ex2.cgi](#)

Dieses Script gibt alle "name-value" Paare der QUERY_STRING Variable aus (GET Methode).

Mein Name: **Textfield:**

Mein Status: Student Mitarbeiter Professor

Dieses Script gibt alle "name-value" Paare von stdin aus (POST Methode).

Mein Name: **Textfield:**

Mein Status: Student Mitarbeiter Professor

Perl mit Mail

Beispiele mit dem Script [ex3.cgi](#) ([mail](#))

Dieses Script schickt eine Email an den angegebenen Adressaten.

E-Mail Adresse: **Textfield:**

Beispiele mit dem Script [ex4.cgi](#) ([logfile](#))

Perl mit Log-Datei

Dieses Script schreibt eine Nachricht in ein Log-File.

Mein Name: **Textfield:**

Mein Status: Student Mitarbeiter Professor

29.5. CGI.pm Modul

Stellt Hilfsmittel für den Einsatz als CGI Programm zur Verfügung.

- Aufruf / Import mit
`use CGI qw/:standard/`
- Erzeugen eines neuen CGI Objekts mit
`$var = CGI -> new( param )`
- Abspeichern eines CGI Objekts in einer Datei
`$var -> save( FILEHANDLE )`
im Format

```
NAME1=VALUE1  
NAME2=VALUE2  
NAME3=VALUE3  
=
```

- Erzeugen eines neuen CGI Objekts aus einer Datei
`$var = CGI -> new( FILEHANDLE )`
- Zugriff auf alle CGI Parameter mit
`VAR -> param( PARAM )`
Die Dekodierung der CGI Parameter erfolgt automatisch.
- Für fast alle HTML Elemente gibt es eine Methode
 - `header()`
 - `start_html()`
 - `end_html()`
 - `p(inhalt)`
 - `textarea(inhalt)`
 - `method({-ATTRIBUT => VALUE}, inhalt)`

Diese Methoden können in beliebigen print-Statements verwendet werden.

Beispiel Gästebuch

In einer Datei werden die Bemerkungen der Gäste zusammen mit ihrem Namen und der Uhrzeit des Eintrags gespeichert.

1. Speichern des aktuellen Eintrags in \$inhalt
2. Einlesen der alten Einträge aus Datei in \$inhalt
3. Abspeichern der Einträge in Datei
4. Aufbau des Eingabeformulars
5. Anzeige der bisherigen Einträge

Gästebuch in Aktion.

```
#!/usr/bin/perl

use strict;
use CGI qw/:standard *table/;
use Fcntl qw/:flock/;

sub fehler {
    my $err = "@_";
    print "\n", h1("Unerwarteter Fehler"), "\n", p($err), "\n", end_html;
}

my(
    $GBuch,    # Name der Gästebuch-Datei
    $MaxGB,    # Maximale Anzahl von Einträgen in GB
    $GBTitel,  # Titel des Gästebuchs
    $akt,      # neuer Eintrag
    @inhalt,   # Inhalt des Gästebuchs
    $eintrag   # ein Eintrag im Gästebuch
);

$GBTitel = "Gästebuch von Heinz Kredel";
$GBuch   = "/tmp/gb-kredel ";
$MaxGB   = 10;

print header;
print start_html({-bgcolor=>'white'}, $GBTitel);
print "\n", h1($GBTitel), "\n";

$akt = CGI->new();

if ($akt->param('eingabe')) {
    $akt->param("datum", scalar localtime);
    @inhalt = ($akt);
}

open(GBUCH, "+< $GBuch") || fehler("$GBuch kann nicht geöffnet werden.");
flock(GBUCH, LOCK_EX) || fehler("$GBuch kann nicht exklusiv verwendet werden");
while (!eof(GBUCH) && @inhalt < $MaxGB) {
    $eintrag = CGI->new(\*GBUCH);
    push @inhalt, $eintrag;
}

seek(GBUCH, 0, 0) || fehler("$GBuch kann nicht zurückgesetzt werden.");
foreach $eintrag (@inhalt) {
    $eintrag->save(\*GBUCH);
}
truncate(GBUCH, tell(GBUCH));
close(GBUCH);

print "\n", hr, "\n", start_form();
print start_table();
print "<tr><td valign='top'> <br>Nachricht: </td>\n<td valign='top'>",
    $akt->textarea(-NAME => "eingabe",
                  -OVERRIDE => 1,
                  -ROWS => "4",
                  -COLS => "50",
                  -VALUE => "" ),
    "</td></tr>\n";
print "<tr><td>Name: </td>\n<td>",
    $akt->textfield(-NAME => "name",
                  -OVERRIDE => 1,
                  -SIZE => "50",
                  -VALUE => $akt->param('name')),
    "</td></tr>\n";
print "<tr><td>", $akt->reset("Löschen"), "</td>\n<td>",
    $akt->submit("Eintragen"), "</td></tr>\n";
print end_table();
```

```
print "\n", end_form(), "\n", hr, "\n";
print h2("Bisherige Einträge"), "\n";
foreach $eintrag (@inhalt) {
    print p($eintrag->param("eingabe")), "\n";
    print p({-ALIGN => "right"},
            $eintrag->param("name"),
            " @ ",
            $eintrag->param("datum"),
            ), "\n";
}
print hr, "\n", end_html, "\n";
```

29.6. Zusammenfassung und Ausblick

- Perl ist nach wie vor eine wichtige Programmiersprache für CGI
- viele Module und Hilfsprogramme für CGI verfügbar
- Apache Modul `perl_module` und `fastcgi_module` erlauben hochperformante CGI Programme
- mit Embedded Perl gibt es eine Konkurrenz zu PHP

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:37:15 CEST 2004

30. PHP Hypertext Preprocessor (PHP)

- Einleitung
- PHP Überblick
- PHP Sprache
- PHP Pakete
- Beispiele



30.1. Einleitung

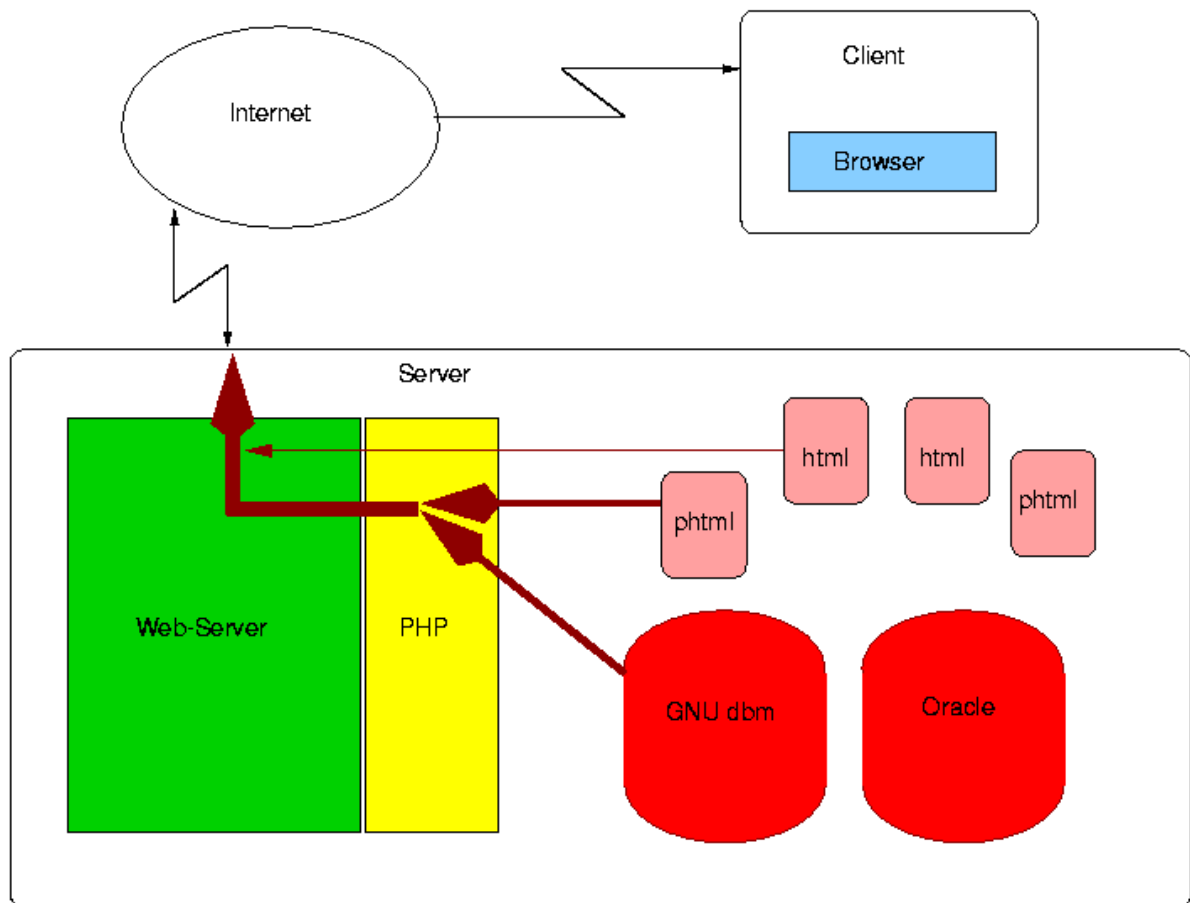
- Server Side Includes (SSI)
- LiveWire
von Netscape
- **PHP/FI 2.0:**
Personal Home Pages / Forms Interpreter
- **PHP 3.0:**
PHP Hypertext Preprocessor
- aktuell: **PHP 4.0:**
PHP Hypertext Preprocessor mit Zend
höhere Performance durch Kompilation
HTTP Sessions eingebaut
bessere Portabilität, auch für MS
- Vergleichbare Ansätze:
ASP Active Server Pages (MS)
JSP Java Server Pages

30.2. PHP Überblick

- CGI Programm oder
- Apache Modul
- gute Datenbank Unterstützung
- Adabas D
- dBase
- mSQL
- MySQL
- Solid
- GNU dbm
- iODBC

PHP Hypertext Preprocessor (PHP)

- Oracle
- Sybase
- Einbettung von Programmen in HTML
- aber auf der Seite des Web-Servers
- die Clients sehen nur noch die Ergebnisse
- als SGML Anwendung:
`<?php ...Programm... ?>`
- als Script:
`<script language="php">...</script>`
- bei Livewire:
`<server>...</server>`
- bei ASP und JSP:
`<% ... %>`



Hello World

Einbettung:

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

```
<title>Test</title>
</head><body>
Test auf PHP:
<?php echo "<p>Hallo von PHP</p>"; ?>
<?php echo "<p>$HTTP_USER_AGENT</p>"; ?>
<?php show_source("hallo.phtml"); ?>
</body>
```

Ausgabe:

```
Test auf PHP:
Hallo von PHP
Mozilla/4.5 [en] (X11; I; Linux 2.0.35 i586)
...
```

[PHP Beispiele](#)

30.3. PHP Sprache

Die Syntax ist von C, Java und Perl abgeleitet.

Werte, Variablen, Typen

- Konversion von Zahlen und Strings ist von Kontext abhängig.
- explizite Anpassung mit Casts: (int), (double), (string).
- Keine Deklarationspflicht für Variablen. aber

```
static $x = "Hallo !\n";
local $x = "Hallo !\n";
global $x;
```
- Operatoren und Ausdrücke wie in C. `y += x--`
- Zugriff auf Umgebungsvariablen etc.
`$GLOBALS['HTTP_REFERER']`
- In neuen Versionen von PHP (ab 4.2.x) ist defaultmässig kein Zugriff auf CGI Variablen mit `$cgi_name` mehr möglich, es geht nur noch

```
$_GET['cgi_name'] für GET Variablen
$_POST['cgi_name'] für POST Variablen
$_REQUEST['cgi_name'] für GET und POST Variablen
$_SERVER['HTTP_REFERER'] für andere Variablen, wie auch QUERY_STRING
```

```
$foo = "0";
$foo++;
$foo += 1;

$bar = (string) $foo;
$foo = (int) $bar;

$a = "b";
$$a = "c";
echo $b;
```

Kontrollstrukturen

- Statements, Expressions, { Statement-Folge }
- if-Statement

```
if (condition1)
```

```
    statement1
[ elseif (condition2)
    statement2 ]
[ else
    statement3 ]
```

```
if (condition):
endif
```

- for-Statement

```
for ([initial-expression]; [condition];
    [increment-expression])
    statement
```

- while-Statement

```
while (condition) {
    statement
}
```

```
while (condition):
endwhile
```

```
do {
    statement
} while (condition)
```

- Zum Beenden von Schleifen break
- Wahrheitswerte wie in Perl.
TRUE: Zahl ungleich 0, nicht-leere Zeichenketten, nicht-leere Objekte
und FALSE: 0, "", leere Objekte
- PHP Programme und HTML Texte dürfen sich Überlappen.

```
<?php if ($a == 9): ?>
<p>Text falls der Wert 9 ist.</p>
<?php endif; ?>
```

Funktionen.

```
function tuwas($a) {
    echo "Eingabe = ", $a;
}
```

```
tuwas("mit einem Text");
```

```
function tuwas($a) {
    return "Eingabe = " . $a;
}
```

```
echo tuwas('mit einem Text');
```

30.4. PHP Pakete

- Adabas D
- Arrays, Felder
- BC, beliebig genaue Arithmetik
- Kalender

- Datum, Zeit
- dBase
- dbm
- Directories, Verzeichnisse
- Aufrufe externer Programme
- filePro
- Filesystem, Dateisystem
- HTTP, Cookies
- Bildbearbeitung, -erzeugung
- IMAP, Email
- Informationen über PHP
- LDAP Verzeichnisdienst
- Mathematische Funktionen
- mSQL
- MySQL
- Sybase
- Netzwerk, Sockets
- ODBC
- Oracle
- PostgreSQL
- Regular Expressions
- Solid
- SNMP
- Strings, Zeichenketten
- URL Bearbeitung
- Datentypen

dbm Datenbank (PHP3)

Ursprünglich von Berkeley, db, gdbm.

Auf (fast) allen Unix Systemen ohne Installation verfügbar.

nur (Key, Value)-Paare

- Datenbank Identifikation

```
int dbmopen(filename, "rwn")
dbmclose(db-identifier)
```
- Zugriff auf Werte

```
bool dbmexists(db-identifier, key)
string dbmfetch(db-identifier, key)
bool dbmdelete(db-identifier, key)
string dbmfirstkey(db-identifier)
string dbmnextkey(db-identifier, key)
```
- Einfügen von Werten

```
bool dbminsert(db-identifier, key, value)
bool dbmreplace(db-identifier, key, value)
```

DBA (Database abstraction layer) (PHP4)

dbm-artige API für PHP4

unterstützt mehrere reale Implementierungen: dbm, ndbm, gdbm, db2, db3, cdb

Auf (fast) allen PHP Systemen existiert eine der DBs

wie dbm nur (Key, Value)-Paare

- Datenbank Identifikation
`int dba_open(filename, "rwnc", "handler")`
`handler = dbm, ndbm, gdbm, db2, db3, cdb` siehe `phpinfo()`
`dba_close(db-handle)`
- Zugriff auf Werte
`bool dba_exists(key, db-handle)`
`string dba_fetch(key, db-handle)`
`bool dba_delete(key, db-handle)`
`string dba_firstkey(db-handle)`
`string dba_nextkey(db-handle)`
- Einfügen von Werten
`bool dba_insert(key, value, db-handle)`
`bool dba_replace(key, value, db-handle)`
- Erweiterungen gegenüber dbm
`int dba_popen(filename, "rwnc", "handler")`
`p=persistent, handler wie oben`
`bool dba_optimize(db-handle)`
`bool dba_sync(db-handle)`

30.5. Beispiel: Zugriffszähler (PHP3)

Einbettung mittels `auto_prepend`, oder direkt in die Seite

Verwendung

```
<?php
/* $counter_start="9999"; */
echo "<h3>" . counter(). " Zugriffe, ";
echo " " . modified("de") . "</h3>";
?>
```

Counter Funktion

```
$filename=$SCRIPT_FILENAME;
$counter_start="1";
function counter() {
    global $filename, $counter_start;
    $counter_dir="/tmp/";
    $counter_db=$counter_dir . "zaehler.dbm";
    if (file_exists("$counter_db")) {
        $db=dbmopen($counter_db,"w");
        if (dbmexists($db,$filename) ) {
            $cnt=dbmfetch($db,$filename);
            if ($counter_start=="1") { $cnt++; }
            else { $cnt=$counter_start; }
            dbmreplace($db,$filename,$cnt);
        }
        else {
            $cnt=$counter_start;
            dbminsert($db,$filename,$cnt);
        }
        dbmclose($db);
        return "$cnt";
    }
}
```

```
    }
    else {
        echo "Attempt to create file: " . $counter_db;
        $cnt=$counter_start;
        $db=dbmopen($counter_db,"n");
        dbminsert($db,$filename,$cnt);
        dbmclose($db);
        return "$cnt";
    }
}
```

Modified Funktion

```
$filename=$SCRIPT_FILENAME;
function modified($lang) {
    global $filename;
    $lm=date("d M Y H:i:s",filectime($filename));
    if ($lang=="de") {
        return "Geändert am $lm";
    } else {
        return "Last modified: $lm";
    }
}
```

30.6. Beispiel: Zugriffszähler (PHP4)

Verwendung wie oben

Counter Funktion

```
$filename=$SCRIPT_FILENAME;
$counter_start="1";
function counter() {
    global $filename, $counter_start;
    $counter_dir="/tmp/";
    $counter_db=$counter_dir . "zaehler.dbm";
    if (file_exists("$counter_db")) {
        $db=dba_open($counter_db,"w","gdbm");
        if ( dba_exists($filename,$db) ) {
            $cnt = dba_fetch($filename,$db);
            if ($counter_start=="1") { $cnt++; }
            else { $cnt=$counter_start; }
            dba_replace($filename,$cnt,$db);
        }
        else {
            $cnt=$counter_start;
            dba_insert($filename,$cnt,$db);
        }
        dba_close($db);
        return "$cnt";
    }
    else {
        echo "Attempt to create file: " . $counter_db;
        $cnt=$counter_start;
        $db=dba_open($counter_db,"n","gdbm");
        dba_insert($filename,$cnt,$db);
        dba_close($db);
        return "$cnt";
    }
}
```

Modified Funktion wie oben

[PHP Beispiele](#)

Installation, Konfiguration

PHP Hypertext Preprocessor (PHP)

Auf jedem Web-Server as CGI Programm

zum Beispiel bei Apache:

```
AddType application/x-php3-script .phtml
Action application/x-php3-script /cgi-bin/php/
```

bei Apache als ladbares Modul

Aktivierung einzelner Datenbanken und Pakete

Weitere Direktiven z.B. in `php3.ini` oder in Apache Konfigurationsfiles mit Prefix `php3_` oder im Code

- Automatisches Hinzufügen
 `auto_append_file` string
 `auto_prepend_file` string
- Fehleranzeige
 `display_errors` bool
 `error_log` string
- Reihenfolge der Variablen, Get, Post, Cookies
 `gpc_order` string, z.B. "GPC"
 `track_vars` bool

Beispiel: [php3.ini](#), [php4.ini](#)

PHP 4 Apache Direktiven:

```
php_value [PHP directive name] [value]
php_flag [PHP directive name] [On|Off]
php_admin_value [PHP directive name] [value]
php_admin_flag [PHP directive name] [On|Off]
```

Sicherheitsprobleme

sollten mit neuen Versionen behoben sein

- Auslieferung von Dateien
 `http://host/cgi_bin/php?/etc/passwd`
 `http://host/cgi_bin/php/etc/passwd`
- Redirect
 `--enable-force-redirect`
- Setzen der Verzeichnisse
 `doc_root`, `user_root`
- Auslagern des Interpreters
 `#!/usr/local/bin/php`

30.7. Ausblick

- LiveWire
 - Servlets
 - Java Server Pages (JSP)
-

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Wed Jul 7 14:44:07 CEST 2004

31. Web-Datenbanken: MySQL

- Einleitung
- MySQL Überblick
- (My)SQL Sprache
- PHP Schnittstelle
- Java JDBC Schnittstelle



31.1. Einleitung

- Datenbanksysteme DBMS
- Relationale Datenbanksysteme RDBMS
- Structured Query Language (SQL), in 1970
- Datenbanken und das Web
- LAMP = Linux + Apache + MySQL + PHP
- WAMP
- LAMPS = LAMP + SSL
- Alternativen: mSQL, Solid, Postgres (PostgreSQL), Oracle, etc.

MySQL Historie

- vor 1994 Postgres (mit PostQUEL) in Nachfolge zu Ingres
Entwickelt als Universitätsprojekt von Michael Stonebreaker
- mSQL als SQL-Interface zu Postgres, jetzt eigene DB-Engine,
Entwickler David Hughes
- Mai 1995 MySQL mit mSQL Interface und eigener DB-Engine (UNIREG B+ ISAM, seit 1979),
Entwickler Michael Widenius

31.2. MySQL Überblick

Features

- hohe Performance, Multithreaded
- Plattform unabhängig
- Entry Level SQL92 Unterstützung

insbesondere:

SELECT, Joins, WHERE-Klausel, GROUP BY

Erweiterungen:

AUTO_INCREMENT, DROP column, REPLACE

in neueren Versionen (ab 3.23.34):

Transaktionen, Commit/Rollback bei Berkeley DB Tabellen

keine Unterstützung:

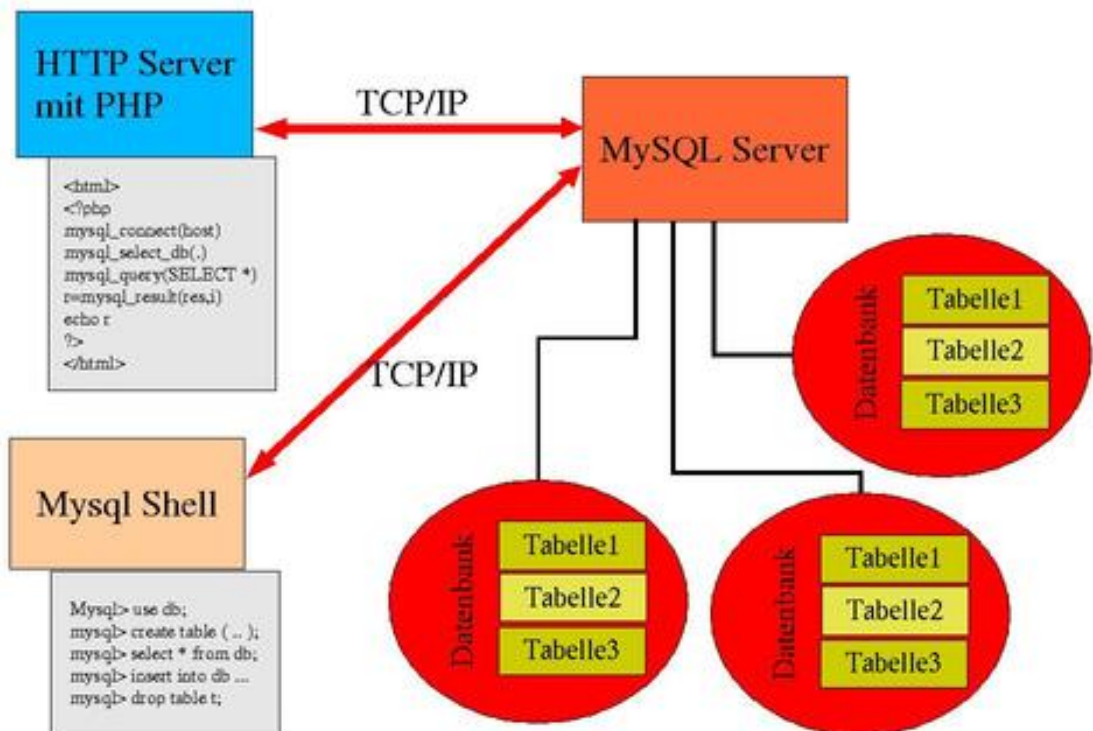
Views, Foreign Keys, Sub-SELECTs

Unterschiede:

GRANT, CREATE/DROP INDEX

- viele APIs, ODBC, JDBC
- ISO8859_1 (Latin1) Unterstützung
- nur der Weiterverkauf kostet Lizenzgebühren
- Zugang zur Datenbank nur über TCP/IP
- eigenes Sicherheitssystem

MySQL Architektur



Installation

- Binär-Versionen oder Source-Code
- jede Datenbank befindet sich in einem eigenen Unterverzeichnis

- jede DB Tabelle befindet sich in 3 Dateien (*.frm, *.ISM, *.ISD)
- Sicherheitssystem in DB 'mysql'
- Perl Interface
- PHP und Apache Zugang
- MySQL gibt es z.B. bei SuSE (fast) fertig installiert

```
wierum{admin}[/usr/local/mysql]516: bin/mysqladmin ver
bin/mysqladmin Ver 6.9 Distrib 3.21.33b, for sun-solaris2.5.1 on sparc
TCX Datakonsult AB, by Monty

Server version          3.21.33b-log
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 11 days 3 hours 13 min 45 sec

Running threads: 2  Questions: 56093  Opened_tables: 19
Flush tables: 1  Open tables: 13
```

Sicherheitssystem

- Script: mysql_install_db
- 'user'-Tabelle regelt Zugriff auf MySQL-Server
Aufbau: (host, user, password, privs, ...)

```
mysql> select host, user, password from user;
+-----+-----+-----+
| host      | user  | password |
+-----+-----+-----+
| localhost | admin | 5f6b52670x3f4407 |
| warum     | admin | 5a6b5y67023f4c07 |
| localhost | gast  |             |
| %         | kredel | 5f6b526c023f4407 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

Benutzer 'gast' darf sich von 'localhost' ohne Passwort verbinden. Die Zugriffsprivilegien sind alle negativ. Benutzer 'kredel' darf sich von überall (%) mit Passwort verbinden.

- 'db'-Tabelle regelt Zugriff auf Datenbanken
Aufbau: (host, db, user, privs, ...)

```
mysql> select host, user, db from db;
+-----+-----+-----+
| host      | user  | db      |
+-----+-----+-----+
| %         |       | test    |
| %         |       | test\_% |
| localhost | gast  | webtech |
+-----+-----+-----+
3 rows in set (0.01 sec)
```

Benutzer 'gast' darf von 'localhost' auf die Datenbank 'webtech' zugreifen. Die Zugriffsprivilegien sind bis auf 'Select_priv' alle negativ.

- 'host'-Tabelle erlaubt Zugriff von Rechnern oder Netzen
Aufbau: (host, db, privs, ...)

```
mysql> select * from host;
Empty set (0.01 sec)
```

Es sind keine Regeln definiert.

- Das Tool mysqlaccess prüft die definierten Rechte.

```
wierum{admin}[/usr/local/mysql]515: bin/mysqlaccess localhost gast webtech
```

```
mysqlaccess Version 2.03, 27 Feb 1997
By RUG-AIV, by Yves Carlier (Yves.Carlier@rug.ac.be)
This software comes with ABSOLUTELY NO WARRANTY.
+++USING FULL WHERE CLAUSE+++
+++USING FULL WHERE CLAUSE+++
+++USING FULL WHERE CLAUSE+++

Access-rights
for USER 'gast', from HOST 'localhost', to DB 'webtech'
+-----+-----+-----+
| Select_priv | Y | | Shutdown_priv | N |
| Insert_priv | N | | Process_priv  | N |
| Update_priv | N | | File_priv     | N |
| Delete_priv | N | | Grant_priv    | N |
| Create_priv | N | | References_priv | N |
| Drop_priv  | N | | Index_priv    | N |
| Reload_priv| N | | Alter_priv    | N |
+-----+-----+-----+

BEWARE: Everybody can access your DB as user `gast' from host `localhost'
        : WITHOUT supplying a password.
        : Be very careful about it!!

The following rules are used:
db      : 'localhost','webtech','gast','Y','N','N','N','N','N','N','N','N','N'
host    : 'Not processed: host-field is not empty in db-table.'
user    : 'localhost','gast','','N','N','N','N','N','N','N','N','N','N','N','N'

BUGs can be reported by email to Yves.Carlier@rug.ac.be
```

MySQL Tools

mysqld, safe_mysqld

MySQL server daemon und Start-Script

mysqlshow

Anzeige diverser Informationen

mysql

SQL shell (GNU readline)

mysqladmin

Administrations Tool, z.B. Start, Stop, Refresh, Create/Drop DB

mysqlaccess

Tool zum Testen und Anzeigen der Sicherheitskonfiguration

mysql_install_db

Initialisierung des Sicherheitssystems

isamchk

Reparatur-Tool für Datenbanken

mysqlshow

mysqlshow Databases

```
wierum{admin}[/usr/local/mysql]502: bin/mysqlshow
+-----+
| Databases |
+-----+
| infoad    |
| mysql     |
| news      |
| rum       |
| stichwort |
| test      |
| ub        |
```

```
| uni      |
+-----+
```

mysqlshow Database: stichwort

```
wierum{admin}[/usr/local/mysql]503: bin/mysqlshow stichwort
Database: stichwort
+-----+
| Tables |
+-----+
| links  |
+-----+
```

mysqlshow Database: stichwort Table: links

```
wierum{admin}[/usr/local/mysql]504: bin/mysqlshow stichwort links
Database: stichwort Table: links Rows: 145
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| link  | char(255) | YES  |     |          |            |
| name  | char(255) | YES  |     |          |            |
| hit   | int(11)   | YES  |     |          |            |
| ID    | int(11)   |      | PRI | 0        | auto_increment |
| ename | char(255) | YES  |     |          |            |
| elink | char(255) | YES  |     |          |            |
+-----+-----+-----+-----+-----+-----+
```

mysql

show tables

```
mysql> show tables from stichwort;
+-----+
| Tables in stichwort |
+-----+
| links               |
+-----+
1 row in set (0.00 sec)
```

show columns

```
mysql> show columns from links from stichwort;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| link  | char(255) | YES  |     | NULL    |            |
| name  | char(255) | YES  |     | NULL    |            |
| hit   | int(11)   | YES  |     | NULL    |            |
| ID    | int(11)   |      | PRI | 0        | auto_increment |
| ename | char(255) | YES  |     | NULL    |            |
| elink | char(255) | YES  |     | NULL    |            |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

show databases

```
mysql> show databases;
+-----+
| Database |
+-----+
| infoad   |
| mysql    |
| news     |
| rum      |
| stichwort |
+-----+
```

```
| test |
| ub   |
| uni  |
+-----+
8 rows in set (0.01 sec)
```

phpMyAdmin

Mit phpMyAdmin existiert ein funktionales Web-Interface zur Verwaltung der MySQL Datenbanken.

[Beispiele](#)

31.3. (My)SQL Sprache

MySQL unterstützt ANSI SQL92 mit den schon genannten Ausnahmen.

SQL Grundkonstrukte

- SELECT zu Auslesen von Daten aus der DB.

```
mysql> select count(*) from links;
+-----+
| count(*) |
+-----+
|      145 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> select sum(hit) from links;
+-----+
| sum(hit) |
+-----+
|     9672 |
+-----+
1 row in set (0.02 sec)
```

```
mysql> select name, hit from links where id="88";
+-----+-----+
| name                | hit |
+-----+-----+
| Philosophische Fakultät | 41  |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> select id, name, hit from links order by hit desc limit 10;
+-----+-----+-----+
| id | name                | hit |
+-----+-----+-----+
| 21 | Bibliothek          | 275 |
| 130 | Universitätsbibliothek | 275 |
| 19 | Beurlaubung          | 231 |
| 20 | Bewerbungsverfahren | 231 |
| 38 | Exmatrikulation      | 231 |
| 58 | Immatrikulation      | 231 |
| 93 | Prüfungsverwaltung   | 231 |
| 100 | Rückmeldung          | 231 |
| 146 | Zulassungsverfahren  | 231 |
| 35 | Email-Verzeichnis    | 215 |
+-----+-----+-----+
10 rows in set (0.01 sec)
```

- INSERT zum Einfügen *neuer* Zeilen in eine Tabelle.

```
mysql> insert into links (name, hit) values ("Fakultät VWL", "1");
Query OK, 1 row affected (0.03 sec)
```

- UPDATE zum Ändern von Werten in den Zeilen.

```
mysql> update links set name="Fakultaet VWL" where id="149";
Query OK, 1 row affected (0.05 sec)
```

- DELETE zum Löschen von Zeilen.

```
mysql> delete from links where id="149";
Query OK, 1 row affected (0.00 sec)
```

SQL Administrationskonstrukte

- (My)SQL Datentypen, z.B. INT(d), CHAR(m), FLOAT, DATE, TIME, VARCHAR(m), TEXT, BLOB.
- CREATE TABLE zum Anlegen von neuen Tabellen.

```
mysql> create table xlink (
        id int(11) not null,
        name char(255),
        link char(255),
        primary key (id) );
Query OK, 0 rows affected (0.15 sec)
```

```
mysql> show columns from xlink from stichwort;
```

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	0	
name	char(255)	YES		NULL	
link	char(255)	YES		NULL	

3 rows in set (0.04 sec)

- DROP TABLE zum Löschen von existierenden Tabellen.

```
mysql> drop table xlink;
Query OK, 0 rows affected (0.10 sec)
```

31.4. PHP Schnittstelle (API)

Das MySQL API ist stark an das mSQL API angelehnt. MySQL unterstützt auch ODBC (Open Database Connectivity) und JDBC (Java Database Connectivity).

Id = mysql_pconnect(hostname, username [, password])

Aufbau einer *persistenten* Verbindung zu einem MySQL Server

Id = mysql_connect(hostname, username [, password])

Aufbau einer Verbindung zu einem MySQL Server

mysql_close(Id)

Schliessen der Verbindung zum MySQL Server Id

Fehler = mysql_error(Id)

Beschreibung des letzten Fehlers beim Zugriff zum MySQL Server Id

DBid = mysql_select_db(database [, Id])

Auswahl einer Datenbank auf einem MySQL Server Id

Res = mysql_query(query [, Id])

Stellen einer Anfrage an einen MySQL Server Id

Res = mysql_db_query(database, query [, Id])

Stellen einer Anfrage an eine Datenbank auf einem MySQL Server Id

num = mysql_num_rows(Res)

Anzahl der Zeilen der Anfrage Res

col = mysql_num_fields(Res)

Anzahl der Spalten (Felder) der Anfrage Res

Data = mysql_result(Res, zeile [, spalte])

Entnahme einer Zeile (oder von Teilen einer Zeile) aus der Anfrage Res

Array = mysql_fetch_row(Res)

Entnahme der nächsten Zeile aus der Anfrage Res

num = mysql_data_seek(Res, zeile)

Positioniert einen Zeiger auf eine Zeile der Anfrage Res

Beispiel:

```
<?php
Function dbQuery ($statement) {
    global $dbconfig;
    mysql_pconnect($dbconfig["sqlserver"],
                  $dbconfig["sqlusername"],
                  $dbconfig["sqlpassword"]);
    mysql_select_db($dbconfig["defaultdb"]);
    $result=@mysql_query($statement);
    if (mysql_error()) { PrintError(mysql_error()); }
    return $result;
}
?>
```

PHP Beispiele

[PHP Stichwortindex](#)

Die Datenbank "webtech" besteht aus einer Tabelle "zaehler" mit zwei Spalten "file" und "count".

```
CREATE TABLE `zaehler` (
  `file` char(200) default NULL,
  `count` int(11) default NULL
) TYPE=ISAM PACK_KEYS=1;
```

[MySQL PHP Counter](#)

31.5. Java Schnittstelle JDBC

Das Java API *Java Database Connectivity* (JDBC) ist nicht an das mSQL API angelehnt, sondern erweitert das ODBC (Open Database Connectivity) API.

JDBC Versionen ab Java 1.4:

1.0 und 1.1:

in etwa die Funktionalität von ODBC, Package `java.sql`

2.1 Core in SE:

Navigation in Resultsets, Batch-Updates, Package `java.sql`

2.0 Optional in EE:

Connection Pooling, Distributed Transactions, neue Datenquellen, Package `javax.sql`

3.0 ab JDK 1.4:

vereinigt 2.1 Core und 2.0 Optional, mehrfache Resultsets, Pooled Statements, XML (Schema) Support, besserer BLOB und CLOB Support, Packages `java.sql`, `javax.sql`, `javax.sql.RowSet`

Die wichtigsten Klassen und Methoden aus `java.sql` sind:

Class.forName(mysqlDriver)

laden des MySQL Treibers: `com.mysql.jdbc.Driver`

DriverManager.getConnection(db,user,password)

erzeugen eines `Connection` Objekts zur Datenbank `db = "jdbc:mysql://host/db"`

dbCon.close()

schliessen der Verbindung zu `dbCon`-Connection

stmt = dbCon.createStatement()

erzeugen eines SQL-Statements zur `dbCon`-Connection

rs = stmt.executeQuery(query)

durchführen der SQL-Abfrage `query` und erzeugen des `ResultSet` `rs`

pstmt = dbCon.prepareStatement(query)

vorbereiten eines SQL-Statements zur `dbCon`-Connection, Parameterpositionen werden durch `?` gekennzeichnet

pstmt.setString(i, param);

pstmt.setInt(i, param);

einsetzen eines Parameters an die `i`. Stelle in dem SQL-Statement

rs = pstmt.executeQuery()

durchführen des vorbereiteten Statements und erzeugen des `ResultSet` `rs`

r = pstmt.executeUpdate()

durchführen eines vorbereiteten Statements und zählen der veränderten Tabellen Zeilen in `r`

stmt.close()

schliessen / freigeben des Statements `stmt`

rs.next()

prüfen, ob das `ResultSet` weitere Zeilen enthält und positionieren des "Cursors" auf die nächste Zeile

rs.getString(i);

rs.getInt('name');

entnehmen des Werts der `i`. Spalte, bzw. der Spalte 'name' aus der aktuellen Zeile

rs.close()

schliessen / freigeben des `ResultSets` `rs`

dbCon.setAutoCommit(false)

einschalten der Transaktionsverwaltung für die `dbCon`-Connection

dbCon.commit()

positives Beenden einer Transaktion, bestätigen aller Änderungen

dbCon.rollback()

negatives Beenden einer Transaktion, verwerfen aller Änderungen

SQLException

wird von fast allen Methoden ausgelöst, wenn etwas schief geht

JDBC Beispiele

Die Datenbank "inet" besteht aus einer Tabelle "counter" mit zwei Spalten "url" und "count".

```
CREATE TABLE `counter` (  
  `url` varchar(255) NOT NULL default '',  
  `count` int(11) NOT NULL default '0',  
  KEY `url` (`url`)  
) TYPE=MyISAM;
```

[MySQL JDBC Counter](#)

[MySQL JDBC Counter Add](#)

31.6. Schlussbemerkungen

- Welche Datenbank ist geeignet?

einfache Anforderungen:

dbm, mSQL, etc.

mittlere Anforderungen:

MySQL, Solid, PostgreSQL, etc.

maximale Anforderungen:

Oracle, DB2, Informix, Sybase, etc.

- Empfehlungen:
nur SQL verwenden
ggf. ODBC Interface verwenden

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat Jun 5 20:57:21 CEST 2004

32. HTTP over SSL (HTTPS)

- S-HTTP
 - Apache mod_ssl
-

32.1. S-HTTP

Secure Hypertext Transfer Protocol

- Definition neuer Elemente in HTML Sprache
- Methoden
- Unterschrift
- Datenverschlüsselung
- Authentifizierung

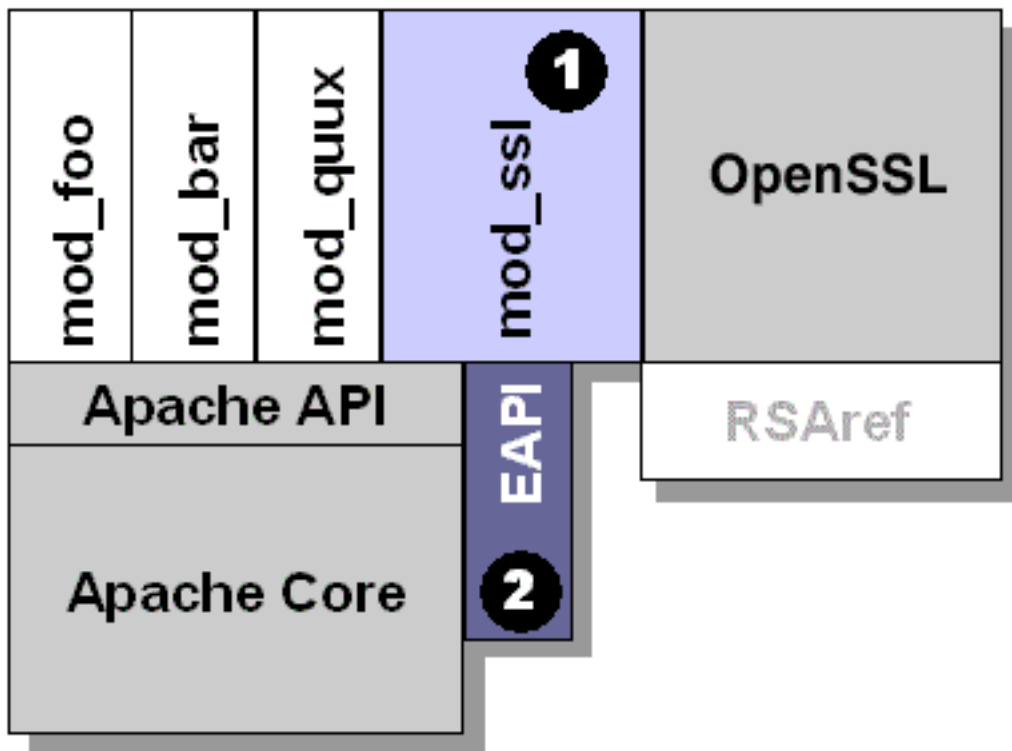
Vorteile

flexibel, sehr viele Verfahren unterstützt
keine Probleme mit Firewalls
erweiterbar ohne Rücksicht auf TCP/IP

Nachteile

nur für HTTP anwendbar
benötigt spezielle Browser Unterstützung

32.2. Apache mod_ssl



`mod_ssl` wurde von von Ralf S. Engelschall im April 1998 entwickelt.

Weiterentwicklung des Apache-SSL Packets von Ben Laurie.

Verwendet OpenSSL von Eric A. Young and Tim Hudson.

Implementiert den `https`: Dienst.
Default Port ist 443.

Konfiguration

für Apache, `mod_ssl`, OpenSSL der SuSE Linux Distribution

1. Sicherstellen, dass alle benötigten Teile installiert sind
Apache, `mod_ssl`, OpenSSL
2. Erzeugen eines Zertifikats
`cd /usr/share/doc/packages/mod_ssl`
`./certificate.sh`
siehe Beispiel
3. Aktivieren von `mod_ssl` in der Apache Konfiguration
in `conf/httpd.conf`
`SSLEngine on`
und einstellen der weiteren erforderlichen Parameter, siehe Beispiel
4. Apache restarten
5. Log Files überwachen
6. SSL Zugriff auf beliebige Seiten mit
`https://host/path/x.html`

Erzeugen eines Zertifikats

1. Kryptographischen Algorithmus auswählen
2. entsprechenden Schlüssel erzeugen
3. Distinguished Name festlegen
damit kann eine Zertifikatanforderung generiert werden
4. erzeugen eines Spiel-Zertifikats
5. schützen des generierten Schlüssels durch Pass-Phrase

```
localhost:/usr/share/doc/packages/mod_ssl # ./certificate.sh
SSL Certificate Generation Utility (mkcert.sh)
Copyright (c) 1998 Ralf S. Engelschall, All Rights Reserved.
```

```
Generating test certificate signed by Snake Oil CA [TEST]
WARNING: Do not use this for real-life/production systems
```

```
STEP 0: Decide the signature algorithm used for certificate
The generated X.509 CA certificate can contain either
RSA or DSA based ingredients. Select the one you want to use.
Signature Algorithm ((R)SA or (D)SA) [R]:
```

```
STEP 1: Generating RSA private key (1024 bit) [server.key]
4139358 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....++++++
.+++++
e is 65537 (0x10001)
```

```
STEP 2: Generating X.509 certificate signing request [server.csr]
Using configuration from .mkcert.cfg
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Country Name (2 letter code) [XY]:DE
2. State or Province Name (full name) [Snake Desert]:Germany
3. Locality Name (eg, city) [Snake Town]:Weinheim
4. Organization Name (eg, company) [Snake Oil, Ltd]:Home
5. Organizational Unit Name (eg, section) [Webserver Team]:Web Lab
6. Common Name (eg, FQDN) [www.snakeoil.dom]:kredel-wh.uni-mannheim.de
7. Email Address (eg, name@FQDN) [www@snakeoil.dom]:kredel@rz.uni-mannheim.de
```

```
STEP 3: Generating X.509 certificate signed by Snake Oil CA [server.crt]
```

```
Certificate Version (1 or 3) [3]:
```

```
Signature ok
```

```
subject=/C=DE/ST=Germany/L=Weinheim/O=Home/OU=Web Lab/CN=kredel-wh.uni-mannheim.de/Email=kredel@rz.uni-man
```

```
Getting CA Private Key
```

```
Verify: matching certificate & key modulus
```

```
read RSA key
```

```
Verify: matching certificate signature
```

```
/etc/httpd/ssl.crt/server.crt: OK
```

```
STEP 4: Encrypting RSA private key with a pass phrase for security [server.key]
```

```
The contents of the server.key file (the generated private key) has to be
kept secret. So we strongly recommend you to encrypt the server.key file
with a Triple-DES cipher and a Pass Phrase.
```

```
Encrypt the private key now? [Y/n]: n
```

```
Warning, you're using an unencrypted RSA private key.
```

```
Please notice this fact and do this on your own risk.
```

```
RESULT: Server Certification Files
```

- o `conf/ssl.key/server.key`
The PEM-encoded RSA private key file which you configure with the 'SSLCertificateKeyFile' directive (automatically done when you install via APACI). KEEP THIS FILE PRIVATE!
- o `conf/ssl.crt/server.crt`
The PEM-encoded X.509 certificate file which you configure with the 'SSLCertificateFile' directive (automatically done when you install via APACI).
- o `conf/ssl.csr/server.csr`
The PEM-encoded X.509 certificate signing request file which you can send to an official Certificate Authority (CA) in order to request a real server certificate (signed by this CA instead of our demonstration-only Snake Oil CA) which later can replace the `conf/ssl.crt/server.crt` file.

WARNING: Do not use this for real-life/production systems

Zum Anzeigen und Bearbeiten der Zertifikate können die OpenSSL Tools verwendet werden.

Apache Konfiguration

wesentlich SSL Engine on

```
<VirtualHost _default_:443>

# General setup for the virtual host
DocumentRoot "/usr/local/httpd/htdocs"
ServerName kredel-wh.isdn.uni-mannheim.de
ServerAdmin root@kredel-wh.isdn.uni-mannheim.de
ErrorLog /var/log/httpd/error_log
TransferLog /var/log/httpd/access_log

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
#SSLEngine off
SSLEngine on

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
SSLCipherSuite ALL:!ADH:!EXP56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. A test
# certificate can be generated with 'make certificate' under
# built time. Keep in mind that if you've both a RSA and a DSA
# certificate you can configure both in parallel (to also allow
# the use of DSA ciphers, etc.)
SSLCertificateFile /etc/httpd/ssl.crt/server.crt
#SSLCertificateFile /etc/httpd/ssl.crt/server-dsa.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/httpd/ssl.key/server.key
#SSLCertificateKeyFile /etc/httpd/ssl.key/server-dsa.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
```

```
# certificate for convinience.
#SSLCertificateChainFile /etc/httpd/ssl.crt/ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCACertificatePath /etc/httpd/ssl.crt
#SSLCACertificateFile /etc/httpd/ssl.crt/ca-bundle.crt

# Certificate Revocation Lists (CRL):
# Set the CA revocation path where to find CA CRLs for client
# authentication or alternatively one huge file containing all
# of them (file must be PEM encoded)
# Note: Inside SSLCARevocationPath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/httpd/ssl.crl
#SSLCARevocationFile /etc/httpd/ssl.crl/ca-bundle.crl

# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

# Access Control:
# With SSLRequire you can do per-directory access control based
# on arbitrary complex boolean expressions containing server
# variable checks and other lookup directives. The syntax is a
# mixture between C and Perl. See the mod_ssl documentation
# for more details.
#<Location />
#SSLRequire (
#    %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
#    and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#    and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#    and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#    and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20
#    or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#) \
#</Location>

# SSL Engine Options:
# Set various options for the SSL engine.
# o FakeBasicAuth:
#   Translate the client X.509 into a Basic Authorisation. This means that
#   the standard Auth/DBMAuth methods can be used for access control. The
#   user name is the 'one line' version of the client's X.509 certificate.
#   Note that no password is obtained from the user. Every entry in the user
#   file needs this password: 'xxj31ZMTZzkVA'.
# o ExportCertData:
#   This exports two additional environment variables: SSL_CLIENT_CERT and
#   SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#   server (always existing) and the client (only existing when client
#   authentication is used). This can be used to import the certificates
#   into CGI scripts.
# o StdEnvVars:
#   This exports the standard SSL/TLS related 'SSL_*' environment variables.
#   Per default this exportation is switched off for performance reasons,
#   because the extraction step is an expensive operation and is usually
#   useless for serving static content. So one usually enables the
#   exportation for CGI and SSI requests only.
# o CompatEnvVars:
#   This exports obsolete environment variables for backward compatibility
#   to Apache-SSL 1.x, mod_ssl 2.0.x, Sioux 1.0 and Stronghold 2.x. Use this
#   to provide compatibility to existing CGI scripts.
# o StrictRequire:
#   This denies access when "SSLRequireSSL" or "SSLRequire" applied even
```

HTTP over SSL (HTTPS)

```
# under a "Satisfy any" situation, i.e. when it applies access is denied
# and no other module can change it.
# o OptRenegotiate:
# This enables optimized SSL connection renegotiation handling when SSL
# directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars +StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/usr/local/httpd/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
# This forces an unclean shutdown when the connection is closed, i.e. no
# SSL close notify alert is send or allowed to received. This violates
# the SSL/TLS standard but is needed for some brain-dead browsers. Use
# this when you receive I/O errors because of the standard approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed, i.e. a
# SSL close notify alert is send and mod_ssl waits for the close notify
# alert of the client. This is 100% SSL/TLS standard compliant, but in
# practice often causes hanging connections with brain-dead browsers. Use
# this only for browsers where you know that their SSL implementation
# works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
SetEnvIf User-Agent ".MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0

# Per-Server Logging:
# The home of a custom SSL log file. Use this when you want a
# compact non-error SSL logfile on a virtual host basis.
CustomLog /var/log/httpd/ssl_request_log \
    "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

Überwachung der Log Files

Den Ablauf einer Verbindung kann man sich mit einem Tool aus OpenSSL anzeigen lassen.

```
openssl s_client -connect localhost:443 -state -debug
GET / HTTP/1.0
```

Der Parameter `s_client` baut eine Verbindung zu einem SSL Server auf, `s_server` baut eine Verbindung zu einem SSL Client auf.

Startup SSL Engine log

```
[29/Oct/2000 19:03:09 12827] [info] Server: Apache/1.3.12, Interface: mod_ssl/2.6.5, Library: OpenSSL/0.9.7
[29/Oct/2000 19:03:09 12827] [info] Init: 1st startup round (still not detached)
[29/Oct/2000 19:03:09 12827] [info] Init: Initializing OpenSSL library
[29/Oct/2000 19:03:09 12827] [info] Init: Loading certificate & private key of SSL-aware server kredel-wh
[29/Oct/2000 19:03:09 12827] [info] Init: Seeding PRNG with 136 bytes of entropy
[29/Oct/2000 19:03:09 12827] [info] Init: Generating temporary RSA private keys (512/1024 bits)
[29/Oct/2000 19:03:09 12827] [info] Init: Configuring temporary DH parameters (512/1024 bits)
```

HTTP over SSL (HTTPS)

```
[29/Oct/2000 19:03:11 12828] [info] Init: 2nd startup round (already detached)
[29/Oct/2000 19:03:11 12828] [info] Init: Reinitializing OpenSSL library
[29/Oct/2000 19:03:11 12828] [info] Init: Seeding PRNG with 136 bytes of entropy
[29/Oct/2000 19:03:11 12828] [info] Init: Configuring temporary RSA private keys (512/1024 bits)
[29/Oct/2000 19:03:11 12828] [info] Init: Configuring temporary DH parameters (512/1024 bits)
[29/Oct/2000 19:03:11 12828] [info] Init: Initializing (virtual) servers for SSL
[29/Oct/2000 19:03:11 12828] [info] Init: Configuring server kredel-wh.isdn.uni-mannheim.de:443 for SSL p

[29/Oct/2000 19:05:11 12841] [info] Connection to child 0 established (server kredel-wh.isdn.uni-mannheim
[29/Oct/2000 19:05:11 12841] [info] Seeding PRNG with 1160 bytes of entropy
[29/Oct/2000 19:06:41 12841] [info] Connection: Client IP: 127.0.0.1, Protocol: SSLv3, Cipher: RC4-MD5 (1
[29/Oct/2000 19:06:41 12841] [info] Initial (No.1) HTTPS request received for child 0 (server kredel-wh.i
[29/Oct/2000 19:06:41 12841] [info] Connection to child 0 closed with standard shutdown (server kredel-wh.i
[29/Oct/2000 19:06:41 12896] [info] Connection to child 1 established (server kredel-wh.isdn.uni-mannheim
[29/Oct/2000 19:06:41 12896] [info] Seeding PRNG with 1160 bytes of entropy
[29/Oct/2000 19:06:41 12907] [info] Connection to child 2 established (server kredel-wh.isdn.uni-mannheim
[29/Oct/2000 19:06:41 12907] [info] Seeding PRNG with 1160 bytes of entropy
[29/Oct/2000 19:06:42 12841] [info] Connection to child 0 established (server kredel-wh.isdn.uni-mannheim
[29/Oct/2000 19:06:42 12841] [info] Seeding PRNG with 1160 bytes of entropy
[29/Oct/2000 19:06:42 12896] [info] Connection: Client IP: 127.0.0.1, Protocol: SSLv3, Cipher: RC4-MD5 (1
[29/Oct/2000 19:06:42 12896] [info] Initial (No.1) HTTPS request received for child 1 (server kredel-wh.i
[29/Oct/2000 19:06:42 12907] [info] Connection: Client IP: 127.0.0.1, Protocol: SSLv3, Cipher: RC4-MD5 (1
[29/Oct/2000 19:06:42 12907] [info] Initial (No.1) HTTPS request received for child 2 (server kredel-wh.i
[29/Oct/2000 19:06:42 12841] [info] Connection: Client IP: 127.0.0.1, Protocol: SSLv3, Cipher: RC4-MD5 (1
[29/Oct/2000 19:06:42 12841] [info] Initial (No.1) HTTPS request received for child 0 (server kredel-wh.i
[29/Oct/2000 19:06:42 12896] [info] Subsequent (No.2) HTTPS request received for child 1 (server kredel-w
[29/Oct/2000 19:06:42 12908] [info] Connection to child 3 established (server kredel-wh.isdn.uni-mannheim
[29/Oct/2000 19:06:42 12908] [info] Seeding PRNG with 1160 bytes of entropy
[29/Oct/2000 19:06:42 12908] [info] Connection: Client IP: 127.0.0.1, Protocol: SSLv3, Cipher: RC4-MD5 (1
[29/Oct/2000 19:06:43 12908] [info] Initial (No.1) HTTPS request received for child 3 (server kredel-wh.i
```

SSL Request log

```
[29/Oct/2000:19:06:41 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET / HTTP/1.0" 1493
[29/Oct/2000:19:06:42 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET /gif/gl.gif HTTP/1.0" 427
[29/Oct/2000:19:06:42 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET /gif/awlogo.gif HTTP/1.0" 12706
[29/Oct/2000:19:06:42 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET /gif/apache_logo.gif HTTP/1.0" 23439
[29/Oct/2000:19:06:42 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET /gif/apache_pb.gif HTTP/1.0" 2326
[29/Oct/2000:19:06:43 +0100] 127.0.0.1 SSLv3 RC4-MD5 "GET /gif/suse_150.gif HTTP/1.0" 2874
```

32.3. Bemerkungen

- Obige Konfiguration nicht in Produktionsumgebungen verwenden!
- Lizenz, Export, Patent Bestimmungen beachten.
- Netzwerk von Zertifikaten aufbauen bzw. anschliessen.
- Problem der Distribution der Zertifikate an die Clients.
- Schutz des Server Key durch Pass-Phrase
(Problem auto restart)
- Schutz des Servers

Erstellt unter Verwendung der Apache mod_ssl Dokumentation.

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:38:00 CEST 2004

33. Java - Applets und Servlets

- Applets
 - Servlets und JSP
 - Apache Tomcat
 - Zusammenfassung und Ausblick
-

33.1. Applets

- kleine eigenständige Anwendung
- kann in Web-Browsern mit JVM ausgeführt werden
- Programmteile werden übers Netzgeladen
- Nutzung der Vorteile der Java Programmiersprache
- Nutzung aller APIs von Java soweit aus Sicherheitsgründen erlaubt
- Sandkasten Prinzip
- insbesondere sind TCP/IP Verbindungen mit eigener Verschlüsselung möglich
- Applet Klasse ist von einer AWT Klasse Component abgeleitet
- AWT definiert Zeichenfunktionen, Buttons, etc.
- Applet definiert zusätzlich Ausführungslogik: init(), start(), stop(), destroy()

Java Applet APIs

```
package java.applet;

import java.awt.*;
import java.awt.image.ColorModel;
import java.net.URL;
import java.net.MalformedURLException;
import java.util.Hashtable;
import java.util.Locale;

public class Applet extends Panel {

    public boolean isActive()
    public URL getDocumentBase()
    public URL getCodeBase()
    public String getParameter(String name)
    public AppletContext getAppletContext()
    public void resize(int width, int height)
    public void resize(Dimension d)
    public void showStatus(String msg)
    public Image getImage(URL url)
    public Image getImage(URL url, String name)
    public final static AudioClip newAudioClip(URL url)
    public AudioClip getAudioClip(URL url)
    public AudioClip getAudioClip(URL url, String name)
    public String getAppletInfo()
    public Locale getLocale()
    public String[][] getParameterInfo()
    public void play(URL url)
    public void play(URL url, String name)

    public void init()
```

```
    public void start()  
    public void stop()  
    public void destroy()  
}
```

```
public class Panel extends Container ...  
public class Container extends Component  
public class Component extends Object implements ... {  
    public void paint(Graphics g) { }  
    ...  
}
```

Hello World Beispiel

Übergabe des Parameters 'nachricht' und Zeichnung des Textes in 'paint'.

```
import java.awt.*;  
import java.applet.*;  
  
public class HelloWorldApplet extends Applet {  
    String msg = "";  
  
    public void init () {  
        msg = getParameter("nachricht");  
    }  
  
    public void paint(Graphics g) {  
        g.drawString("Hello World ...",10,50);  
        g.drawString(msg,10,75);  
        g.drawString("... vom Applet.",10,100);  
    }  
}
```

Einbettung in HTML mit dem Applet-Element.

```
<applet code="HelloWorldApplet.class" width="150" height="150">  
<param name="nachricht" value="mit Parameter">  
</applet>
```

Beispiele

[Hello World Applet](#)

[Uhren-Demo Applet](#)

[Demo Applets](#)

33.2. Servlets

- Erweiterung der Web-Server Funktionalität
- kann CGI ersetzen, bzw. ergänzen
- Einbindung via Server-API oder Standalone
- Nutzung der Vorteile der Java Programmiersprache
- Nutzung aller APIs von Java

- Apache: JServ, Jakarta Projekt
- Integration in Java-Web-Server: Jigsaw
- Websphere von IBM, JRun von Live Software, etc
- Kombination mit Java Server Pages (JSP)
- z.Z. Version 2.2., vom 17. Dez. 1999
- Web-Application Konzept



Java Servlet APIs

- Basis Funktionalität: Kontextverwaltung, Sende- und Empfangs-Objekte, Service-Methode
- und eine an CGI angelehnte Schnittstelle: Http, Cookies, Sitzungsverwaltung
- Basis Interfaces: Servlet, ServletConfig, ServletContext, ServletRequest, ServletResponse
- Abstrakte Klassen: GenericServlet, ServletInputStream, ServletOutputStream
- HTTP Interfaces: HttpServletRequest, HttpServletResponse, HttpSession, HttpSessionContext
- (Abstrakte) Klassen: HttpServlet, Cookie
- Es wird im Web-Server nur ein Servlet-Objekt erzeugt, nicht für jede Anfrage ein neues
- Beachte Multithreading: service, doGet können parallel aufgerufen werden

Basis Funktionalität

```
package javax.servlet;  
  
public interface Servlet {  
    public void init(ServletConfig config) throws ServletException;  
    public ServletConfig getServletConfig();  
    public void service(ServletRequest req, ServletResponse res)  
        throws ServletException, IOException;  
    public String getServletInfo();  
    public void destroy();  
}
```

```
public interface ServletConfig { }  
  
public interface ServletRequest {  
    public int getContentTypeLength();  
    public ServletInputStream getInputStream() throws IOException;  
    ....  
}
```

```
}  
public interface ServletResponse {  
    public ServletOutputStream getOutputStream() throws IOException;  
    public void setContentType(String type);  
    ...  
}  
  
public abstract class GenericServlet  
    implements Servlet, ServletConfig, java.io.Serializable  
{  
    public void init(ServletConfig config) throws ServletException  
    public abstract void service(ServletRequest req, ServletResponse res)  
        throws ServletException, IOException;  
    ...  
}
```

Hello World Beispiel mit GenericServlet

```
import java.io.*;  
import java.text.*;  
import java.util.*;  
import javax.servlet.*;  
  
public class SimpleHelloWorld extends GenericServlet {  
    private int aufrufe;  
    public void init(ServletConfig config) throws ServletException  
    {  
        super.init(config);  
        aufrufe=0;  
    }  
    public void service(ServletRequest request,  
        ServletResponse response)  
        throws IOException, ServletException  
    {  
        response.setContentType("text/html");  
        ServletOutputStream out = response.getOutputStream();  
  
        out.println("<html>");  
        out.println("<head>");  
  
        String title = "Hello World";  
        aufrufe++;  
  
        out.println("<title>" + title + "</title>");  
        out.println("</head>");  
        out.println("<body bgcolor=\"white\">");  
  
        out.println("<h1>" + title + "</h1>");  
        out.println("<h2>" + aufrufe + " Aufrufe</h2>");  
        out.println("</body>");  
        out.println("</html>");  
    }  
}
```

Erweiterte Funktionalität zu HTTP

```
package javax.servlet.http;  
  
public abstract class HttpServlet extends GenericServlet
```

```
implements java.io.Serializable
{
    public HttpServlet()
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    protected long getLastModified(HttpServletRequest req)
    private void doHead(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    ...
}
```

Hello World Beispiel mit HttpServlet

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldExample extends HttpServlet {
    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");

        String title = rb.getString("helloworld.title");

        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");

        out.println("<h1>" + title + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

33.3. Java Server Pages (JSP)

- Einbettung von Java in HTML
- wie PHP oder ASP
- Kennzeichnung durch `<% Java Code %>`
- aus dem Java Code wird beim ersten Aufruf ein Servlet generiert und dieses dann kompiliert (javac)
- die Ausführung des Byte-Code ist dann wesentlich schneller
- der Web-Server wird konfiguriert, dass Dateien mit bestimmter Endung (*.jsp) von dem JspServlet behandelt werden

- Import von Java Klassen mit
`<%@ page import "java.util.Date" %>`
- Einfügen von anderen Dateien
`<%@ include file="dateiname" %>`
- Zugriff auf Variablen der service-Methode von Servlet
- Einfügen von Variablen und Ausdrücken in HTML mit '='
`<%= var.name; %>`
- Zugriff auf JSP-Teile im XML Stil
`<jsp:expr> request.getQueryString() </jsp:expr>`
- Deklarationen mit '!'
`<%! int xyz = 1; %>`
- Definition eigener Tags
`<%@ taglib="URL" prefix="my" %>`
- Verwendung von Java Beans möglich
`<jsp:usebean id="." class="xyz" >`

Hello World Beispiel mit JSP

```
<html>
<head>
<title>Hello World JSP</title>
</head>

<body bgcolor="white">
<hr>
<%
    out.println("<h2>Hallo Welt! von JSP</h2>");
%>
<hr>
</body>
</html>
```

33.4. Apache Jakarta Tomcat Projekt

- Apache Jakarta
Apache plus diverse Java Aktivitäten
- Apache Jakarta Tomcat
Java Servlet und JSP Teil
- wird offiziell von Sun unterstützt
Referenzimplementierung
- Servlet API 2.2, 2.3 (Februar 2001)
- Java Server Pages 1.1 (Februar 2001), 1.2
- Tomcat 3.0 (1999)
- Tomcat 4.x (Ende 2001): Catalina
- Tomcat 5.x (2004): Servlet-API 2.4, JSP 2.0



Varianten der Installation

1. Standalone
Tomcat stellt einen Web-Server und den Servlet/JSP Server
 2. als Teil von Apache httpd
der httpd führt via mod_jserv die JVM plus Tomcat-Klassen als Subprozess aus
 3. getrennt von Apache httpd
Tomcat stellt einen separaten Prozess für Servlets und JSP
der httpd kommuniziert via mod_jserv und einem TCP/IP basierten Protokoll (AJP V12) mit Tomcat
- Standalone ist gut zum Entwickeln und Testen
ist relativ einfach zu konfigurieren
 - getrennte Ausführung ist performanter für produktiven Einsatz
ist schwer zu Konfigurieren: Teile des Web-Baums werden von Apache selbst behandelt, während nur die Servlet/JSP Teile an Tomcat delegiert werden
Apache (mod_jserv) kann mit vielen Servlet/JSP Servern gleichzeitig umgehen
 - Tomcat kann so auch unabhängig von Apache zusammen mit anderen Web-Servern verwendet werden

Standalone Installation

- es gibt ein tomcat Script (BAT-Datei) zum Starten und Stoppen
- dieses Script muss nur das java Programm (und Tomcat-Home) finden

Installation als Subprozess von Apache

- Das Apache Modul mod_jserv muss installiert sein
- in der Apache Konfiguration (httpd.conf) muss mod_jserv konfiguriert werden

```
LoadModule jserv_module /usr/lib/apache/mod_jserv.so
ApJServManual Off
ApJServProperties /etc/httpd/jserv/jserv.properties
```

- LoadModule aktiviert den Java Server Teil
- ApJServManual Off, d.h. automatisches starten der JVM
- ApJServProperties definiert die Konfigurationsdatei von Tomcat selbst
Definition der JVM und des CLASSPATH für die Tomcat und Servlet Klassen
- mit

```
AddType text/jsp .jsp
AddHandler jserv-servlet .jsp
```

wird der Java Server Pages Handler definiert

Installation separat zu Apache

- Das Apache Modul `mod_jserv` muss installiert sein
- in der Apache Konfiguration (`httpd.conf`) muss `mod_jserv` konfiguriert werden

```
LoadModule jserv_module /usr/lib/apache/mod_jserv.so
ApJServManual On
ApJServMount /servlet ajpv12://localhost:8007/servletzone
```

- `LoadModule` aktiviert den Java Server Teil
- `ApJServManual On`, d.h. manuelles starten der JVM von Tomcat
- `ApJServMount` definiert die URLs (`/servlet`), die an Tomcat weitergereicht werden
`ajpv12://localhost:8007/servletzone`
- `ajpv12`: definiert das Kommunikationsprotokoll zu Tomcat
- `localhost:8007` definiert den Host und den Port an dem Tomcat 'sitzt'
- `servletzone` definiert den Tomcat Bereich (Zone), der die Servlets ausführen soll
- daneben werden mit den normalen Apache Direktiven die Verzeichnisse definiert, die Seiten (HTML, PHP, etc) enthalten, die Apache selbst ausliefert
- mit

```
AddType text/jsp .jsp
AddHandler jserv-servlet .jsp
```

wird der Java Server Pages Handler definiert

Tomcat Konfiguration

1. `server.xml` definiert die Tomcat Basiskonfiguration
2. `web.xml` definiert die Tomcat Grundkonfiguration für Web-Applikationen zusammen mit `web.xml` Dateien für jede Web-Applikation für spezifischen Konfiguration der Applikation

TOMCAT WEB SERVER ADMINISTRATION TOOL

Commit Changes Log Out

Tomcat Server

- Service (Tomcat-Standalone)
 - Connector (8009)
 - Connector (8080)
 - Host (localhost)**
 - Context (/)
 - Context (/admin)
 - Context (/examples)
 - Context (/manager)
 - Context (/tomcat-docs)
 - Context (/velexample)
 - Context (/webdav)
 - Logger for Host (localhost)
 - Logger for Service (Tomcat-Standalone)
 - Realm for Service (Tomcat-Standalone)
- Resources
 - Data Sources
 - Mail Sessions
 - Environment Entries
 - User Databases
- User Definition
 - Users
 - Groups
 - Roles

Host (localhost) Host Actions: -----Available Actions-----

Save Reset

Host Properties

Property	Value
Name:	localhost
Application Base:	webapps
Auto Deploy:	True
Debug Level:	0
Deploy XML:	True
Live Deploy:	True
Unpack WARs:	True

Tomcat Web-Administration

server.xml

Grundaufbau (Tomcat 4.x):

```
<Server>
  <Service>
    <Connector />
    <Engine>
      <Host>
        <Context />
      </Host>
    </Engine>
  </Service>
</Server>
```

Beispiel (Tomcat 4.x):

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">

  <!-- Define the Tomcat Stand-Alone Service -->
  <Service name="Tomcat-Standalone">

    <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8080" minProcessors="5" maxProcessors="75"
      enableLookups="true" redirectPort="8443"
      acceptCount="100" debug="0" connectionTimeout="20000"
      useURValidationHack="false" disableUploadTimeout="true" />

    <!-- Define the top level container in our container hierarchy -->
    <Engine name="Standalone" defaultHost="localhost" debug="0">

      <!-- Global logger unless overridden at lower levels -->
      <Logger className="org.apache.catalina.logger.FileLogger"
        prefix="catalina_log." suffix=".txt"
        timestamp="true"/>

      <!-- Define the default virtual host -->
      <Host name="localhost" debug="0" appBase="webapps"
        unpackWARs="true" autoDeploy="true">

        <!-- Tomcat Examples Context -->
        <Context path="/examples" docBase="examples" debug="0"
          reloadable="true" crossContext="true">
          <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="localhost_examples_log." suffix=".txt"
            timestamp="true"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

- das Element Server definiert einen eignen Tomcat Server
- das Element Service definiert eine Gruppe von zusammen gehörigen Diensten durch Connectoren zu einer Engine
- die Elemente Connector definieren Ports und Klassen für die verschiedenen TCP/IP Verbindungen der Dienste
- das einzige Element Engine definiert einen Container für mehrere virtuelle Hosts
- das Element Logger definiert diverse Protokollierungsoptionen
- die Elemente Host definieren die verschiedenen virtuellen Hosts
- die Elemente Context definieren die Zuordnung zwischen URLs (Zonen, path) und Datei-Pfaden (docBase), sowie deren Optionen (debug, reloadable)

Tomcat Verzeichnisstruktur

- bin Skripte (BAT-Dateien)
- conf Konfigurationsdateien (server.xml, web.xml)
- server Jar-Dateien mit diversen Tomcat Klassen, werden in CLASSPATH von Tomcat (aber nicht der Anwendungen) aufgenommen
- shared zusätzliche Java Klassen und Jar-Dateien, die nur von den Web-Applikationen geteilt werden
- common zusätzliche Java Klassen und Jar-Dateien, die von Tomcat und den Web-Applikationen geteilt werden
- logs Protokolldateien
- webapps die eigentlichen Web-Applikationen: Servlets, HTML-Dateien, JSP-Dateien, War-Dateien
- work Servlets, die zu den JSP Dateien automatisch generiert werden
- doc Dokumentation der APIs etc.

- src Java Code der APIs, soweit benötigt

Aufbau der Web-Applikationen

- befinden sich in einem Unterverzeichnis von webapps
- in diesem Verzeichnis (und ggf. Unterverzeichnissen) befinden sich alle HTML, JSP, etc Dateien der Web-Applikation
- in der Datei WEB-INF/web.xml befindet sich die Konfiguration der Web-Applikation
Parameter, Sicherheitsoptionen, etc.
der sogenannte 'Web Application Deployment Descriptor'
- in dem Unterverzeichnis WEB-INF/classes befinden sich die Servlet Klassen
- in dem Unterverzeichnis WEB-INF/lib befinden sich weitere JAR Dateien der Web-Applikation
- das Verzeichnis kann mit dem Tool ant verwaltet werden
- das Verzeichnis kann in eine sogenannte War-Datei (Web Application Archive) gepackt werden, die sich leicht auf andere Rechner übertragen lässt

web.xml

Aufbauprinzip:

```
<web-app>
  <display-name />
  <context-param />
  <filter />
  <filter-mapping />
  <listener />
  <servlet />
  <servlet-mapping >
    <url-pattern />
  </servlet-mapping >
  <session-config />
  <mime-mapping />
  <welcome-file-list />
  <security-constraint />
</web-app>
```

Die Reihenfolge der Elemente ist signifikant, da die URL-Muster in dieser Reihenfolge getestet werden. Die URL-Muster werden immer relativ zum aktuellen Verzeichnis interpretiert, auch wenn sie mit / beginnen.

Beispiel mit globalen Definitionen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
  <servlet>
    <servlet-name>default</servlet-name>
    <servlet-class>org.apache.tomcat.servlets.DefaultServlet
    </servlet-class>
  </servlet>
  <servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>org.apache.tomcat.servlets.InvokerServlet
    </servlet-class>
  </servlet>
  <servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.runtime.JspServlet
    </servlet-class>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>invoker</servlet-name>
```

```
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>jsp</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>
...
<mime-mapping>
  <extension>txt</extension>
  <mime-type>text/plain</mime-type>
</mime-mapping>
...
</web-app>
```

- das Element web-app definiert die Konfiguration der Applikation
- die Elemente servlet definieren die Zuordnung von Servlet Namen zu Servlet Klassen
- die Elemente servlet-mapping definieren die Zuordnung von Servlet Namen zu URL Mustern
- die Elemente mime-mapping definieren die Zuordnung von Datei-Endungen zu Mime-Types

Beispiel mit applikationsspezifischen Definitionen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
  <servlet>
    <servlet-name>hallo</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
    <init-param>
      <param-name>von</param-name>
      <param-value>Karl Dall</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>hallo</servlet-name>
    <url-pattern>/hallo.html</url-pattern>
  </servlet-mapping>

  <security-constraint>
    ...
  </security-constraint>

  <login-config>
    ...
  </login-config>
</web-app>
```

Die Bedeutung der Elemente ist wie in der globalen web.xml Datei

Wenn sich die Anwendung im Verzeichnis /opt/jakarta/tomcat/webapps/beispiel befindet, bezieht sich /hallo.html auf den URL `http://host:port/beispiel/hallo.html`.

Manager für Applikationen

Tomcat Web Application Manager

Message: OK - Reloaded application at context path /velexample

Manager

List Applications	HTML Manager Help	Manager Help
-------------------	-------------------	--------------

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	1	Start Stop Reload Remove
/admin	Tomcat Administration Application	true	1	Start Stop Reload Remove
/examples	Tomcat Examples	true	0	Start Stop Reload Remove
/manager	Tomcat Manager Application	true	0	Start Stop Reload Remove
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Reload Remove
/velexample		true	0	Start Stop Reload Remove
/webdav	Webdav Content Management	true	0	Start Stop Reload Remove

Tomcat Web-Application Manager (1)

Install

Install directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

Upload a WAR file to install

Select WAR file to upload

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/4.1.27	1.4.2-b28	Sun Microsystems Inc.	Linux	2.4.21-99-default	i386

Tomcat Web-Application Manager (2)

33.5. Zusammenfassung und Ausblick

- ideale Programmiersprache und Entwicklungsumgebung fürs Web
- Applets erweitern die Möglichkeiten der Browser (UAs)
- Servlets erweitern die Möglichkeiten der Web-Server
- Java Beans, Enterprise Java Beans
- JDBC, RMI, CORBA
- Java Web-Applications
- Cocoon
- SOAP

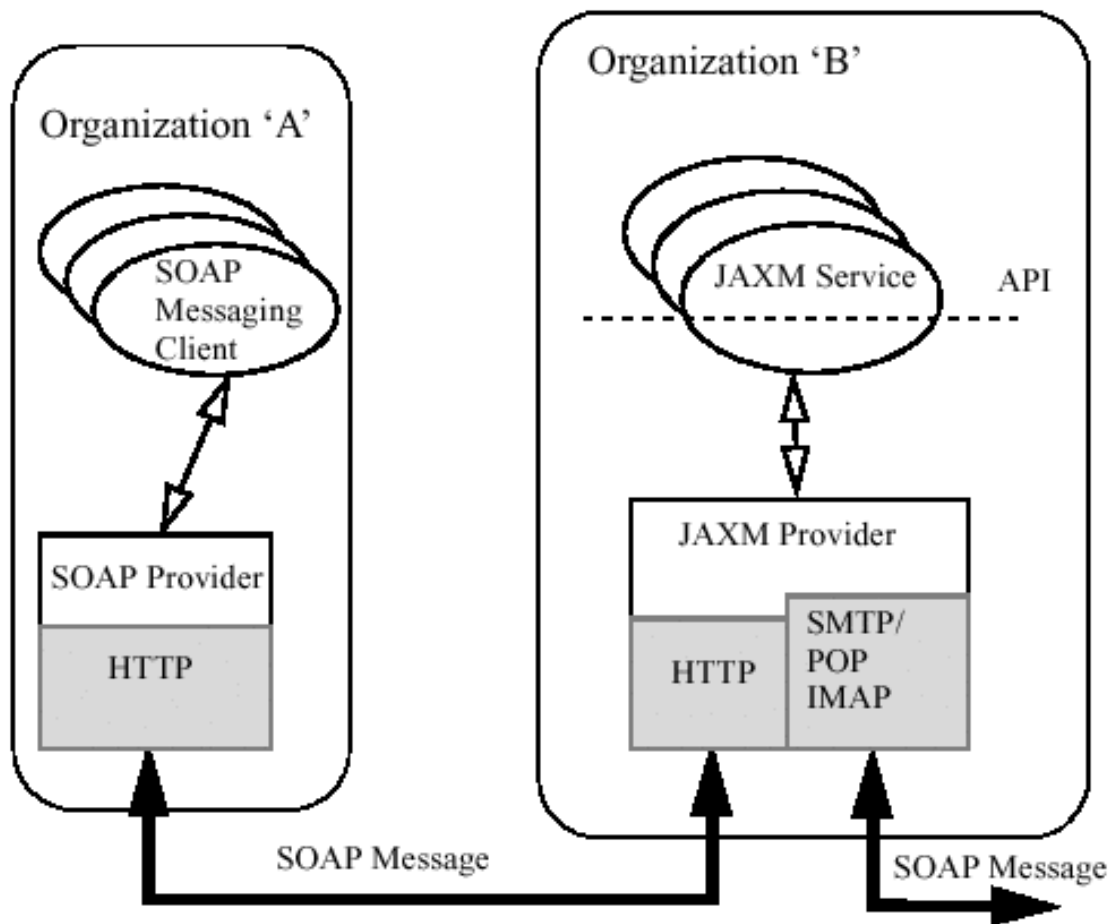
Last modified: Mon May 31 14:47:18 CEST 2004

34. SOAP & WSDL

- Simple Object Access Protocol
 - SOAP Nachrichten
 - SOAP Encoding
 - SOAP und RPC
 - SOAP Attachments
 - Web Service Description Language
 - WSDL Datentypen
 - WSDL Nachrichten
 - WSDL Porttypes
 - WSDL Binding
 - WSDL Service und Port
 - WSDL - SOAP Binding
-

34.1. Simple Object Access Protocol

- Protokoll zum Informationsaustausch (im Web)
- Inhalt der Information wird per XML beschrieben
- definiert 'Envelope' und seine Verarbeitung
- definiert 'Encoding' für die Übertragung
- definiert 'Remote Procedure Call' für Objekt Verarbeitung
- Transport der Information wird per HTTP oder HTTP-EF erledigt
- Vorteil: Strukturierte XML-Daten anstelle von binären MIME-Daten



SOAP Prozessmodell und SOAP - JAXM Zusammenarbeit
(Quelle: JAXM Spec)

'Historie':

- SOAP 1.0 von MS
- SOAP 1.1 ist W3C Note vom 8. Mai 2000
- SOAP Attachments ist W3C Working Draft

Beispiel: Hallo SOAP Server

```
POST /HalloServer HTTP/1.1
Host: www.hallo-welt.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "ein URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetGreeting xmlns:m="ein NS-URI">
      <myName>Heinz Kredel</myName>
    </m:GetGreeting>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Beispiel: Antwort vom SOAP Server

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:getGreetingResponse xmlns:m="ein NS-URI">
      <message>Hallo Heinz Kredel!</message>
    </m:GetGreetingResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Transport in HTTP

- neuer HTTP Header SOAPAction: [action-URI]
- Content-Type ist text/xml
- bei SOAP Fehler muss auch ein HTTP Fehler 500 erzeugt werden und ein Fault-Element muss in der Antwort enthalten sein

Arten von SOAP Nachrichten

- einfache SOAP Nachrichten
- SOAP Nachrichten mit Attachments

34.2. SOAP Nachrichten

verwendete Namensräume

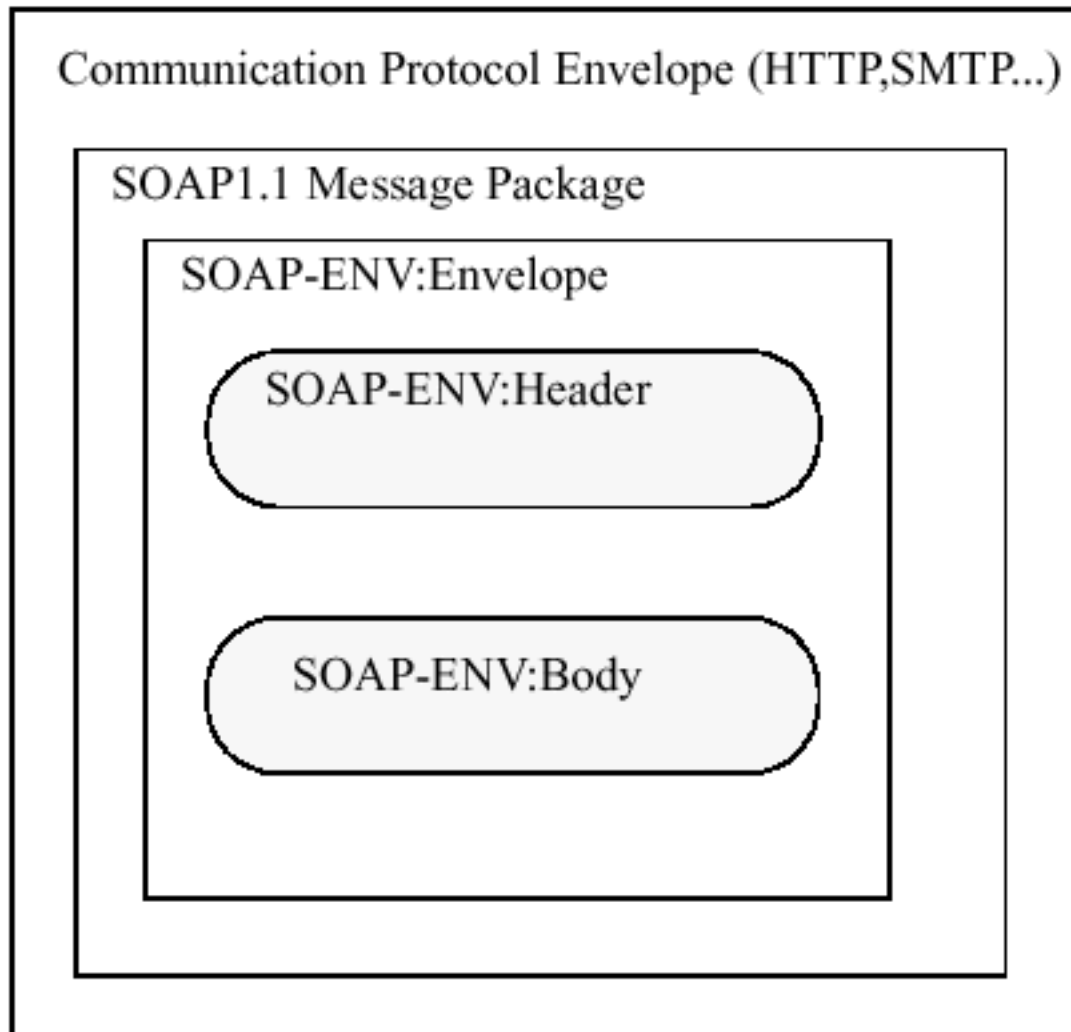
- xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
- xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"

SOAP Nachrichten dürfen keine DTDs (Document Type Definitions) und keine PIs (Processing INstruction) enthalten

Ein SOAP-Processor (Empfänger) muss in der Lage sein

- alle Teile einer SOAP Nachricht zu identifizieren.
- Versteht er einige Teile nicht muss er die Nachricht verwerfen.
- Ist der Inhalt nicht für ihn, muss er seine Teile entfernen und den Inhalt weiterleiten.

SOAP Envelope



SOAP ohne Attachments
(Quelle: JAXM Spec)

- eine SOAP Nachricht ist/besteht aus einem SOAP-Envelope
- ein SOAP-Envelope besteht aus einem (optionalen) SOAP-Header und einem SOAP-Body, danach dürfen weitere (nicht-SOAP) XML-Elemente folgen
- der SOAP-Header dient zur Angabe von speziellen Eigenschaften des SOAP-Bodies
- der SOAP-Body enthält die eigentliche Nachricht für den (nächsten) Empfänger
- es gibt einen default SOAP-Body für Fehlermeldungen: SOAP-ENV:Fault
- der (optionale) Inhalt von SOAP-Header muss mit den richtigen Namensräumen versehen sein
- der Inhalt von SOAP-Body kann(?) mit entsprechenden Namensräumen versehen sein
- XML-Elemente nach SOAP-Body müssen mit den richtigen Namensräumen versehen sein
- alle SOAP Elemente können ein `encodingStyle` Attribut haben
- weitere Attribute für den Inhalt der Header und des Body:
 - `actor` = "URI" definiert den Empfänger des Elements

- `mustUnderstand = "1"` verlangt, dass der Empfänger die Bedeutung des Elements voll versteht, falls nicht, wird die Nachricht verworfen

allgemeinste Form eines SOAP-Envelopes

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "NS-URI"
  encodingStyle = "NS-URI" >
  <SOAP-ENV:Header>
    Header Inhalt
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    Body Inhalt
  </SOAP-ENV:Body>
  <xxx:other xmlns:xxx = "NS-URI" >
    anderer Inhalt
  </xxx:other>
</SOAP-ENV:Envelope>
```

das SOAP-Fault Element

```
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Server</faultcode>
    <faultstring>Beschreibung</faultstring>
    <faultactor>actor-URI</faultactor>
    <detail>
      <e:message xmlns:e="ein NS-URI">
        Kann das Element xxx nicht verarbeiten.
      </e:message>
    </detail>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
```

Fehler-Codes in faultcode sind analog zu HTTP 1xx, 2xx, 3xx, 4xx, 5xx Codes:

- `versionMismatch`: Namensräume passen nicht
- `mustUnderstand`: kann ein solches Element nicht verstehen
- `Client`: die Anfrage war Fehlerhaft, hatte nicht die korrekte Authentifizierung, etc.
- `Server`: der Server konnte die Nachricht nicht verarbeiten oder weiterleiten

34.3. SOAP Encoding

- ein einfaches Typ-System zur Beschreibung der übertragenen Daten
- alles was mit XML Schema definierbar ist kann übertragen werden
- verwendete Schema Namensräume
- `xmlns:xsd = "http://www.w3.org/1999/XMLSchema"`
- `xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"`
- im Zweifelsfall oder für bessere Effizienz können entsprechende 'SOAP-ENC' Attribute den Typ genauer beschreiben
- einfache Datentypen: int, float, String, Enumeration, Byte Arrays; z.B.

```
<betrag xsi:type="xsd:float">29.95</betrag>
oder auch nur
```

```
<betrag>29.95</betrag>
```

- zusammengesetzte Datentypen: Struct, Array; z.B.

```
<zahlenFeld SOAP-ENC:arrayType="xsd:int[2]">  
  <zahl>3</zahl>  
  <zahl>5</zahl>  
</zahlenFeld>
```

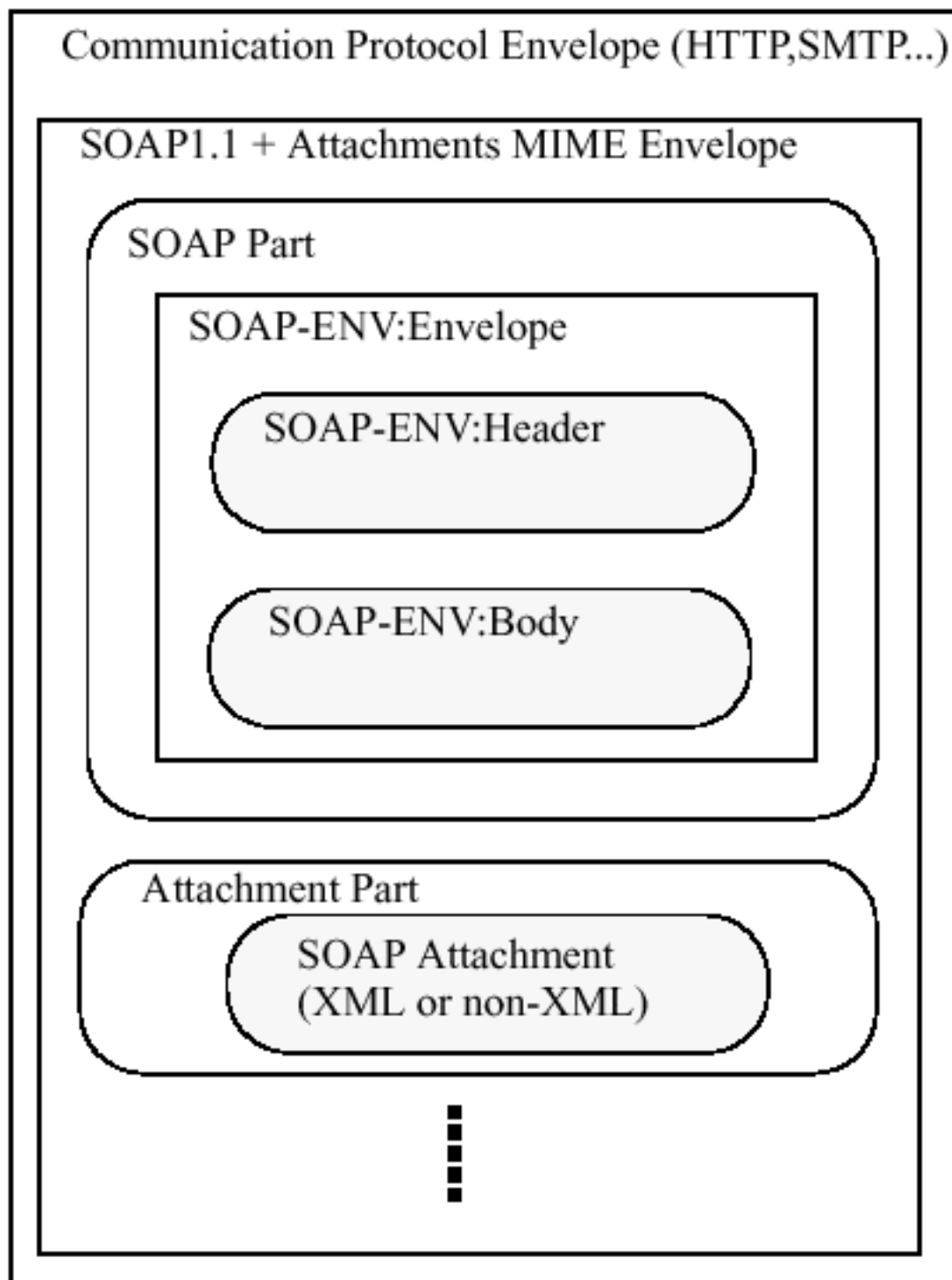
34.4. SOAP für RPC

Funktionsaufrufe werden im SOAP-Body als zusammengesetzter Datentyp übertragen.

z.B.

```
<methodName>  
  <arg1Name>3</arg1Name>  
  <arg2Name>5</arg2Name>  
  <arg3Name>7</arg3Name>  
</methodName>
```

34.5. SOAP mit Attachments

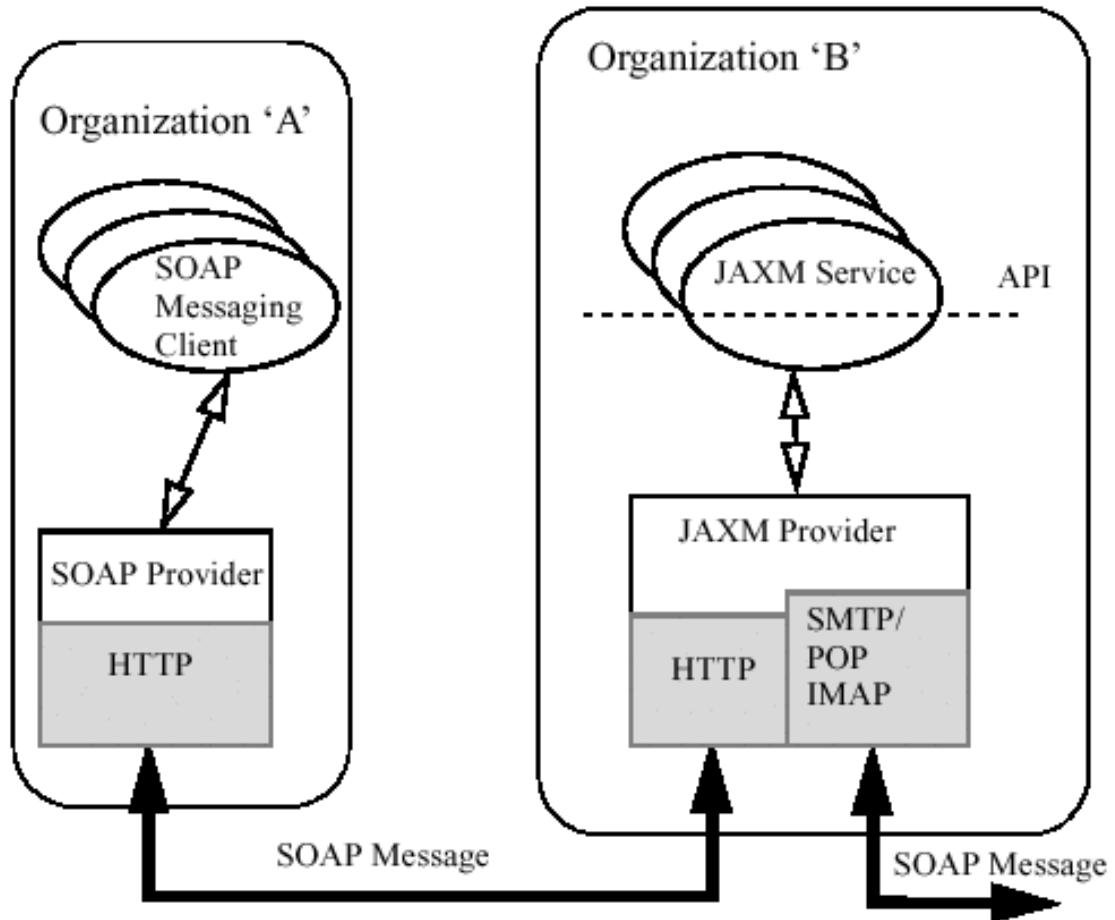


SOAP mit Attachments (Quelle: JAXM Spec)

34.6. JAXM: Java API for XML Messaging

Unterstützt SOAP und SOAP mit Attachments.

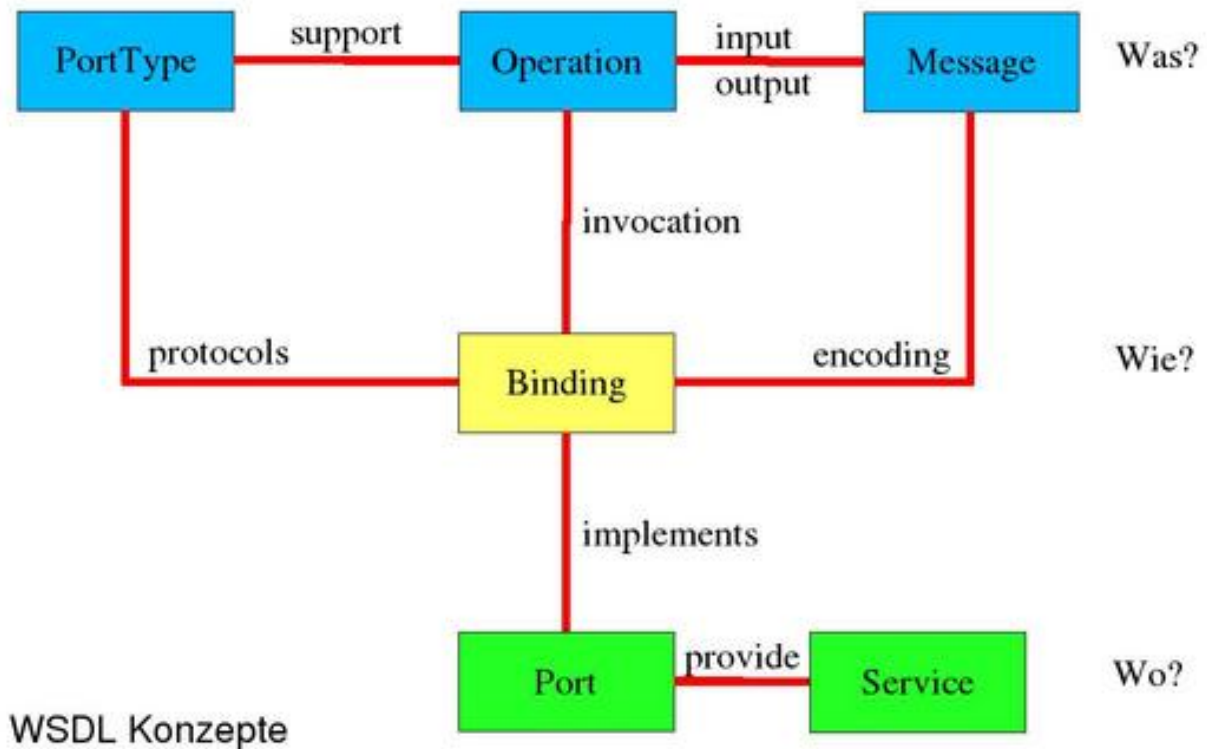
Unterstützt ebXML: e-bussines-XML.



SOAP - JAXM Zusammenarbeit
(Quelle: JAXM Spec)

34.7. Web Service Description Language (WSDL)

- WSDL 1.0 von Microsoft
- WSDL 1.1 ist W3C Note vom 15. März 2001
- WSDL 2.0 ist beim W3C in Arbeit
- Interface Definition von Web-Services
- ähnlich wie Java Interfaces oder CORBA IDLs
- kann RPC- oder Message-Artige Schnittstellen beschreiben
- Beschreibung durch XML-Dokumente



Quelle: Web Services, Kossmann, Leymann, Informatik Spektrum 04.2004

Übersicht über WSDL Konzepte

Was bietet der Web-Service ?

- Definition der Datentypen durch XML-Schema
- Beschreibung von Nachrichten-Typen durch Komposition aus Folgen von Datentypen
- Beschreibung von Operationen durch Komposition von Nachrichten-Typen
- Zusammenfassung von Operationen zu Service-Schnittstellen (PortTypes genannt)
- entspricht etwa einer Java interface oder CORBA IDL interface Definition

Wie werden die Informationen transportiert ?

- Binden der Operationen an Übertragungsprotokolle (z.B. SOAP, HTTP, Mime/SMTP)
- Binden der Nachrichten an Übertragungsmechanismen und (Transport-)Kodierungen
- ist bei Java z.B. RMI, Sockets, JMS, bei CORBA z.B. IIOP

Wo befindet sich der Web-Service ?

- Definition des Web-Service durch Ports mit Transportmethode und URL
- ist bei Java z.B. rmi : -URL und RMI-Registry, bei CORBA z.B. IOR : -URL und Naming-Service

Grundaufbau einer Web-Service Definition:

```
<wsdl:definitions xmlns:wsdl=url >
  <wsdl:types > XML Schema Definitionen </wsdl:types>
  <wsdl:message name=n > Nachrichtentypen </wsdl:message>
  <wsdl:portType name=n > Operationen </wsdl:portType>
  <wsdl:binding name=n > Protokolle und Encoding </wsdl:binding>
  <wsdl:service name=n > Port und URL </wsdl:service>
</wsdl:definitions>
```

Definition der Datentypen

Es können eingene Datentypen definiert werden und es können alle Datentypen von XML Schema verwendet werden.

```
<wsdl:types >
  <xsd:schema >
    <xsd:complexType name=n > ... </xsd:complexType>
    <xsd:simpleType name=n > ... </xsd:simpleType>
    ...
  </xsd:schema >
</wsdl:types>
```

Beispiel zur Definition eines String-Feldes:

```
<wsdl:types>
  <schema
    targetNamespace="http://localhost:8080/axis/CarStore.jws"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
    <complexType name="ArrayOf_xsd_string">
      <complexContent>
        <restriction base="soapenc:Array">
          <attribute ref="soapenc:arrayType"
            wsdl:arrayType="xsd:string[]" />
        </restriction>
      </complexContent>
    </complexType>
  </schema>
</wsdl:types>
```

Definition der Nachrichten

Eine Nachricht ist eine Folge aus einzelnen Datentypen, die durch Parts definiert werden. Eine Nachricht entspricht etwa einem Parameter bei einem Methodenaufruf.

```
<wsdl:message name="n">
  <wsdl:part name="p" element="tns:name" />
  <wsdl:part name="p" type="xsd:type" />
</wsdl:message>
```

Beispiel zur Definition der Nachrichten der folgenden Methode `String getCar( int: i )`:

```
<wsdl:message name="getCarRequest">
  <wsdl:part name="iCar" type="xsd:int" />
</wsdl:message>
<wsdl:message name="getCarResponse">
  <wsdl:part name="getCarReturn" type="xsd:string" />
</wsdl:message>
```

Definition der Operationen

Operationen werden innerhalb des portType Elements definiert. Für jede Operation werden Ein- und Ausgabe Parameter definiert. Es werden folgende 4 Arten unterschieden: One-way, Request-response, Solicit-response, Notification.

```
<wsdl:portType name="n">
  <wsdl:operation name="oneWay" >
    <wsdl:input message="impl:n" name="n"/>
  </wsdl:operation>

  <wsdl:operation name="reqRes" >
    <wsdl:input message="impl:n" name="n"/>
    <wsdl:output message="impl:n" name="n"/>
  </wsdl:operation>

  <wsdl:operation name="solRes" >
    <wsdl:output message="impl:n" name="n"/>
    <wsdl:input message="impl:n" name="n"/>
  </wsdl:operation>

  <wsdl:operation name="not" >
    <wsdl:output message="impl:n" name="n"/>
  </wsdl:operation>
</wsdl:portType>
```

Beispiel zur Definition der Operation, die folgender Methode `String getCar( int: i )` entspricht:

```
<wsdl:portType name="CarStore">
  <wsdl:operation name="getCar" parameterOrder="iCar">
    <wsdl:input message="impl:getCarRequest" name="getCarRequest"/>
    <wsdl:output message="impl:getCarResponse" name="getCarResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

Bindungen zwischen Definitionen und Transport

Mit Hilfe des Binding-Elements wird definiert wie die portTypes über das Netz transportiert und kodiert werden. In WSDL 1.1. sind folgende Transportarten beschrieben: SOAP, direkt mit HTTP GET / POST oder mit Mime / SMTP. Transportstile sind: RPC oder Message / Document.

Mit dem WSDL-SOAP Binding ergibt sich folgender Aufbau:

```
<wsdl:binding name="nBinding" type="impl:">
  <wsdlsoap:binding style="rpc" transport=url />

  <wsdl:operation name="nN">
    <wsdlsoap:operation soapAction="" />

    <wsdl:input name="getCarRequest">
      <wsdlsoap:body encodingStyle=url namespace=url use=u />
    </wsdl:input>

    <wsdl:output name="nN">
      <wsdlsoap:body encodingStyle=url namespace=url use=u />
    </wsdl:output>

    <wsdl:fault name="nN">
      <wsdlsoap:body encodingStyle=url namespace=url use=u />
    </wsdl:fault>

  </wsdl:operation>
</wsdl:binding>
```

Beispiel für das Binding zur Operation, die folgender Methode `String getCar( int: i )` entspricht:

```
<wsdl:binding name="CarStoreSoapBinding" type="impl:CarStore">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="getCar">
```

```
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getCarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="getCarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

Services und Ports

Mit Hilfe des Service-Elements werden verschiedene (Kommunikations-) Endpunkte gruppiert. Ein Endpunkt wird durch ein Port-Element definiert. Der Port stellt die Beziehung zwischen der Bindung und dem URL der Service-Implementierung her.

Mit dem WSDL-SOAP Binding ergibt sich folgender Aufbau:

```
<wsdl:service name="nS">
  <wsdl:port binding="impl:nB" name="nN">
    <wsdlsoap:address location=url />
  </wsdl:port>
</wsdl:service>
```

Beispiel für den Service der einen Autospeicher bereitstellt. Der Autospeicher nutzt die Apache Axis Implementierung von SOAP.

```
<wsdl:service name="CarStoreService">
  <wsdl:port binding="impl:CarStoreSoapBinding" name="CarStore">
    <wsdlsoap:address location="http://localhost:8080/axis/CarStore.jws"/>
  </wsdl:port>
</wsdl:service>
```

WSDL - SOAP Bindungselemente

Die WSDL-SOAP Bindungselemente werden in den entsprechenden WSDL Elementen eingefügt und definieren wie die Elemente in SOAP zu interpretieren sind.

Die wichtigsten Elemente (mit wsdlsoap Namensraum) sind:

```
<wsdlsoap:binding style="rpc|document"
  transport="soap/{http,ftp,smtp}" />
<wsdlsoap:operation soapAction="http header" />
<wsdlsoap:body encodingStyle=url namespace=url
  use="literal|encoded" />
<wsdlsoap:header message="qname"
  encodingStyle=url namespace=url use="literal|encoded" />
<wsdlsoap:headerfault message="qname"
  encodingStyle=url namespace=url use="literal|encoded" />
<wsdlsoap:fault name="qname"
  encodingStyle=url namespace=url use="literal|encoded" />
<wsdlsoap:address location=url />
```

Beispiel für eine SOAP Anfrage und Antwort aus dem Autospeicher Web-Service. Die SOAP Anfrage nach der getCar Operation:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCar soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <arg0 xsi:type="xsd:int">2</arg0>
    </getCar>
  </soapenv:Body>
</soapenv:Envelope>
```

Der Web Service antwortet mit folgender SOAP Nachricht

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCarResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getCarReturn xsi:type="xsd:string">BMW</getCarReturn>
    </getCarResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

WSDL - HTTP Bindungselemente

Die WSDL-HTTP Bindungselemente werden in den WSDL Elementen (binding, operation und address) eingefügt und definieren wie die Elemente im HTTP Protokoll zu interpretieren sind.

Die wichtigsten Elemente (mit `wSDLhttp` Namensraum) sind:

```
<wSDLhttp:binding verb="GET|POST" />
<wSDLhttp:operation location=relative-url />
<wSDLhttp:urlReplacement />
<wSDLhttp:address location=absolute-url />
```

WSDL - MIME Bindungselemente

Die WSDL-MIME Bindungselemente werden in WSDL Elementen (input und output) eingefügt und definieren wie die Elemente als MIME Dokumente zu interpretieren sind.

Die wichtigsten Elemente (mit `wSDlmime` Namensraum) sind:

```
<wSDlmime:content part="n" type="type/subtype" />
<wSDlmime:multipartRelated >
  <wSDlmime:part>
    <wSDlmime:content type="type/subtype" />
  </wSDlmime:part>
  <wSDlmime:part>
    <wSDLsoap:body part="n" use="u" />
  </wSDlmime:part>
</wSDlmime:multipartRelated>
<wSDlmime:mimeXmlcontent part="n" />
```

In der WSDL Spezifikation wird eine XML Schema Definition aller Elemente und Bindungs-Elemente angegeben.

Ausblick

- Universal Description Discovery and Integration (UDDI)
Verzeichnisdienst
 - Grid Services im Grid Computing
 - Grid-WSDL (GWSDL)
-

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sun Jul 4 15:05:16 CEST 2004

35. Apache Axis

- Apache Axis
 - Java Web Service (JWS)
 - Java Web Service Client
 - Web Services Deployment Descriptor (WSDD)
-

35.1. Apache Axis

- die SOAP Engine der Apache Group
- Version 1.1, 2003, aus IBMs SAOP4J hervorgegangen
- Standalone und Servlet Versionen
- unterstützt SOAP 1.1 und 1.2
- umfangreiche WSDL Unterstützung
- Erzeugen von WSDL SOAP-Requests
- Parsen der WSDL SOAP-Responses und Erzeugen entsprechender Java Objekte
- Erzeugen von WSDL Beschreibungen aus Java Klassen
- Erzeugen von Java Klassen(skeletons) aus WSDL Beschreibungen
- Interfaces und Klassen in `org.apache.axis`
- Verwendet JAX-RPC (`javax.xml.rpc`)
- die SOAP Nachrichten werden durch `org.apache.axis.Message` Objekte repräsentiert
- automatisches Deployment und (De-)serialisierung von Java Beans mit Java Web Service (JWS)
- HTTP und JMS Transport
- ein TCP-Monitor zur Analyse der SOAP Nachrichten
- von Sun gibt es ein "Web Service Development Package", das vergleichbares zu Axis enthält

Java Web Service (JWS)

- Dateien mit der Endung `.jws`, die eine JavaBean implementieren, können sofort als Web Service verwendet werden
- `jws`-Dateien werden auf das Axis Servlet gemapped
- eine JavaBean verlangt im wesentlichen einen Konstruktor ohne Argumente und die Zugriffsmethoden müssen als `getXXX()` bzw. `setXXX()` implementiert sein

Ein einfacher Java Web Service: `CarStore.jws`

```
public class CarStore {
    private static String[] cars = null;

    public CarStore() {
        if ( cars != null ) {
            return;
        }
        cars = new String[] {
            "Opel",
            "Porsche",
            "BMW",
        }
    }
}
```

```

        "Audi",
        "VW",
        "Mazda",
        "Toyota",
        "Volvo"
    };
}

public String getCar( int iCar ) {
    if ( cars == null ) {
        return "uninitialized";
    }
    if ( iCar < 0 || cars.length <= iCar ) {
        return "out of bounds";
    }
    return cars[ iCar ];
}

public String getCars( ) {
    if ( cars == null ) {
        return "uninitialized";
    }
    StringBuffer result = new StringBuffer();
    for ( int i = 0; i < cars.length; i++ ) {
        result.append(" " + cars[i] );
    }
    return result.toString();
}

public String[] getCarArray( ) {
    if ( cars == null ) {
        return new String[] { "uninitialized" };
    }
    return cars;
}

public void setAppendCar(String car) {
    String[] newCars = new String[ cars.length + 1];
    for ( int i = 0; i < cars.length; i++ ) {
        newCars[i] = cars[i];
    }
    newCars[ newCars.length -1 ] = car;
    cars = newCars;
    return;
}
}

```

Das AxisServlet stellt mit dem Parameter wsdl (der Aufruf URL ist CarStore.jws?wsdl) folgende WSDL Beschreibung des Service zur Verfügung:

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    targetNamespace="http://localhost:8080/axis/CarStore.jws"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:apache="http://xml.apache.org/xml-soap"
    xmlns:impl="http://localhost:8080/axis/CarStore.jws"
    xmlns:intf="http://localhost:8080/axis/CarStore.jws"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <schema
            targetNamespace="http://localhost:8080/axis/CarStore.jws"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
            <complexType name="ArrayOf_xsd_string">
                <complexContent>
                    <restriction base="soapenc:Array">
                        <attribute ref="soapenc:arrayType"
                            wsdl:arrayType="xsd:string[]"/>
                    </restriction>
                </complexContent>
            </complexType>
        </schema>
    </wsdl:types>

```

```

        </restriction>
    </complexContent>
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="getCarResponse">
    <wsdl:part name="getCarReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="setAppendCarResponse">
</wsdl:message>
<wsdl:message name="getCarArrayRequest">
</wsdl:message>
<wsdl:message name="setAppendCarRequest">
    <wsdl:part name="car" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getCarsRequest">
</wsdl:message>
<wsdl:message name="getCarRequest">
    <wsdl:part name="iCar" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="getCarArrayResponse">
    <wsdl:part name="getCarArrayReturn" type="impl:ArrayOf_xsd_string"/>
</wsdl:message>
<wsdl:message name="getCarsResponse">
    <wsdl:part name="getCarsReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="CarStore">
    <wsdl:operation name="getCar" parameterOrder="iCar">
        <wsdl:input message="impl:getCarRequest" name="getCarRequest"/>
        <wsdl:output message="impl:getCarResponse" name="getCarResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getCarArray">
        <wsdl:input message="impl:getCarArrayRequest" name="getCarArrayRequest"/>
        <wsdl:output message="impl:getCarArrayResponse" name="getCarArrayResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getCars">
        <wsdl:input message="impl:getCarsRequest" name="getCarsRequest"/>
        <wsdl:output message="impl:getCarsResponse" name="getCarsResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setAppendCar" parameterOrder="car">
        <wsdl:input message="impl:setAppendCarRequest" name="setAppendCarRequest"/>
        <wsdl:output message="impl:setAppendCarResponse" name="setAppendCarResponse"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CarStoreSoapBinding" type="impl:CarStore">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getCar">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getCarRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://DefaultNamespace" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="getCarResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
        </wsdl:output>
    </wsdl:operation>
<wsdl:binding name="CarStoreSoapBinding" type="impl:CarStore">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getCar">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getCarRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://DefaultNamespace" use="encoded"/>
        </wsdl:input>
        <wsdl:output name="getCarResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
        </wsdl:output>
    </wsdl:operation>
<wsdl:operation name="getCarArray">
    <wsdlsoap:operation soapAction=""/>

```

```
<wsdl:input name="getCarArrayRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="getCarArrayResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getCars">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="getCarsRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://DefaultNamespace" use="encoded"/>
  </wsdl:input>
  <wsdl:output name="getCarsResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="setAppendCar">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="setAppendCarRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://DefaultNamespace" use="encoded"/>
  </wsdl:input>
  <wsdl:output name="setAppendCarResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://localhost:8080/axis/CarStore.jws" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="CarStoreService">
  <wsdl:port binding="impl:CarStoreSoapBinding" name="CarStore">
    <wsdlsoap:address location="http://localhost:8080/axis/CarStore.jws"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

mit dem Aufruf `CarStore.jws?method=getCar&x=0` erhält man folgende Antwort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCarResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getCarReturn xsi:type="xsd:string">Opel</getCarReturn>
    </getCarResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

mit dem Aufruf `CarStore.jws?method=getCars` erhält man folgende Antwort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCarsResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getCarsReturn
        xsi:type="xsd:string">
        Opel Porsche BMW Audi VW Mazda Toyota Volvo
      </getCarsReturn>
    </getCarsResponse>
  </soapenv:Body>
```

```
</soapenv:Envelope>
```

mit dem Aufruf `CarStore.jws?method=getCarArray` erhält man folgende Antwort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCarArrayResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getCarArrayReturn
        xsi:type="soapenc:Array"
        soapenc:arrayType="xsd:string[8]"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        <item>Opel</item>
        <item>Porsche</item>
        <item>BMW</item>
        <item>Audi</item>
        <item>VW</item>
        <item>Mazda</item>
        <item>Toyota</item>
        <item>Volvo</item>
      </getCarArrayReturn>
    </getCarArrayResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

mit dem Aufruf `CarStore.jws?method=setAppendCar&x=DaimlerChrysler` erhält man folgende Antwort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <setAppendCarResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </soapenv:Body>
  </soapenv:Envelope>
```

der Inhalt des Objekts `CarStore` scheint sich aber nicht zu ändern

Java Web Service Client

- Einsatz der Klassen `Service` und `Call`
- aus dem Paket `org.apache.axis.client`

Ein einfacher Client, der aus dem `CarStore` WS, ein Auto ausliest: `CarStoreGetCar`

```
import java.io.IOException;
import java.net.URL;
import java.net.MalformedURLException;
import java.rmi.RemoteException;

import javax.xml.namespace.QName;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.rpc.ServiceException;

public class CarStoreGetCar {

    final static String carStoreUrl =
        "http://localhost:8090/axis/services/CarStore";
```

```
// or "http://localhost:8090/axis/CarStore.jws";
final static String operation = "getCar";

public static void main(String[] args) {

    Service service = new Service();
    Integer carNo = new Integer( 2 );
    Object[] params = new Object[] { carNo };
    String answer = "";

    try {
        URL url = new URL( carStoreUrl );
        Call call = (Call) service.createCall();

        call.setTargetEndpointAddress( url );
        call.setOperationName( operation );

        answer = (String) call.invoke( params );
        System.out.println("gesuchtes Auto = "
                           + answer);
    } catch(MalformedURLException e) {
        e.printStackTrace();
    } catch(ServiceException e) {
        e.printStackTrace();
    } catch(RemoteException e) {
        e.printStackTrace();
    }
}
}
```

Wird axis/service/CarStore angesprochen muss der Service mit WSDD (siehe unten) beschrieben und deployed worden sein.

Wird axis/CarStore.jws angesprochen, dann wurde der entsprechende Service mit WSDD (siehe unten) automatisch beschrieben und deployed.

Die Ausgabe ist z.B.

```
gesuchtes Auto = BMW
```

Der Client erzeugt folgende SOAP Anfrage

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCar soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <arg0 xsi:type="xsd:int">2</arg0>
    </getCar>
  </soapenv:Body>
</soapenv:Envelope>
```

Der Web Service antwortet mit folgender SOAP Nachricht

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCarResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getCarReturn xsi:type="xsd:string">BMW</getCarReturn>
    </getCarResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Bei der Operation `getCars` erzeugt der Client folgenden SOAP Body in der Anfrage

```
<soapenv:Body>
  <getCars
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</soapenv:Body>
```

und erhält folgende Antwort im SOAP Body

```
<soapenv:Body>
  <getCarsResponse
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <getCarsReturn xsi:type="xsd:string"> Opel Porsche BMW Audi
      VW Mazda Toyota Volvo</getCarsReturn>
  </getCarsResponse>
</soapenv:Body>
```

Bei der Operation `getCarArray` erzeugt der Client folgenden SOAP Body in der Anfrage

```
<soapenv:Body>
  <getCarArray
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
</soapenv:Body>
```

Die Anfrage lässt sich genau so gut an JWS stellen, dabei wird der gleiche SOAP Body wie oben verwendet.

Die main-Methode ist wie folgt zu ändern:

```
final static String carStoreUrl =
    "http://localhost:8090/axis/CarStore.jws";
//    "http://localhost:8090/axis/services/CarStore";
...
    answer = (String[]) call.invoke( params );
    System.out.println("gefundene Autos:");
    for ( int i = 0; i < answer.length; i++ ) {
        System.out.println("Auto[" + i + "] = " + answer[i]);
    }
...
```

Die Antwort ist jetzt ein SOAP String Array

```
<soapenv:Body>
  <getCarArrayResponse
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <getCarArrayReturn
      xsi:type="soapenc:Array"
      soapenc:arrayType="xsd:string[8]"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
      <item>Opel</item>
      <item>Porsche</item>
      <item>BMW</item>
      <item>Audi</item>
      <item>VW</item>
      <item>Mazda</item>
      <item>Toyota</item>
      <item>Volvo</item>
    </getCarArrayReturn>
  </getCarArrayResponse>
</soapenv:Body>
```

Die Ausgabe des Clients ist dann

```
gefundene Autos:
Auto[0] = Opel
```

```
Auto[1] = Porsche
Auto[2] = BMW
Auto[3] = Audi
Auto[4] = VW
Auto[5] = Mazda
Auto[6] = Toyota
Auto[7] = Volvo
```

Bei der Operation `setAppendCar` erzeugt der Client folgenden SOAP Body in der Anfrage

```
<soapenv:Body>
  <setAppendCar
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <arg0 xsi:type="xsd:string">Renault</arg0>
  </setAppendCar>
</soapenv:Body>
```

Die Antwort ist leer, da es sich um eine void-Methode handelt, und sieht wie folgt aus

```
<soapenv:Body>
  <setAppendCarResponse
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  </setAppendCarResponse>
</soapenv:Body>
```

Fragt man dann mit `CarStoreGetCarArray` den Inhalt wieder ab, ergibt sich

```
gefundene Autos:
Auto[0] = Opel
Auto[1] = Porsche
Auto[2] = BMW
Auto[3] = Audi
Auto[4] = VW
Auto[5] = Mazda
Auto[6] = Toyota
Auto[7] = Volvo
Auto[8] = Renault
```

Statische Variablen bleiben also wie gewohnt bis zum Neuladen der Klasse erhalten. Um Informationen dauerhaft zu speichern müssen Datenbanken, Preferences oder Dateien verwendet werden.

Web Services Deployment Descriptor (WSDD)

- Zuordnung von Java Klassen zu WSDL Beschreibungen
- JWS erzeugt automatisch WSDDs und deployed die Anwendung
- für einen Service wird ein Name und ein Provider in dem Element `deployment` definiert
- Provider (auch als Service Style bezeichnet) können z.B. RPC oder Message sein
- der Service benötigt die Parameter `className` und `allowedMethods`
- `className` definiert die Java Klasse, die den Service leistet.
Die Klasse muss sich z.B. in `axis/WEB-INF/classes` befinden.
- `allowedMethods` listet alle (public) Methoden, die der Service nach aussen bereitstellen soll
- die WSDD-Datei wird mit dem Tool `org.apache.axis.client.AdminClient` verarbeitet, das die entsprechenden Eintragungen im Server vornimmt
- Web Services bleiben über den Neustart von Tomcat hinweg deployed und verfügbar
- die Service Objekte werden für jede Anfrage neu erzeugt, die Klassen werden nur bei Bedarf neu geladen, d.h. um Änderungen persistent zu speichern muss z.B. eine Datenbank verwendet werden
- Sollen eigene Datentypen transportiert werden, müssen mit dem service-Subelement `typeMapping` die entsprechenden Zuordnungen vorgenommen werden:
 - ein allgemeiner Name

- die implementierende Java Klasse
- die Klasse, die serialisiert (z.B. BeanSerializerFactory)
- die Klasse, die deserialisiert (z.B. BeanDeserializerFactory)
- und der Encoding-Style

Beispiel für den CarStore: CarStoreUnDeploy.wsdd.

Als Provider ist hier mit java:RPC die RPC-artige Durchführung gewählt.

```
<?xml version="1.0" encoding="utf-8" ?>
<deployment name="test"
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java" >
  <service name="CarStore"
    provider="java:RPC" >
    <parameter name="className"
      value="CarStore" />
    <parameter name="allowedMethods"
      value="*"
      value="getCar getCars setAppendCar getCarArray" />
  </service>
</deployment>
```

Die Aktivierung sieht z.B. wie folgt aus:

```
java ... org.apache.axis.client.AdminClient CarStoreDeploy.wsdd
Processing file CarStoreDeploy.wsdd
<Admin>Done processing</Admin>
```

- zum Entfernen eines Services gibt es einen Undeployment Descriptor
- in dem Element undeployment wird nur der Name des Service angegeben, der entfernt werden soll
- das name Attribut von service spezifiziert den Name des Web Services
- eine Undeployment-WSDD-Datei wird ebenfalls mit dem Tool AdminClient verarbeitet

Beispiel CarStoreUnDeploy.wsdd

```
<?xml version="1.0" encoding="utf-8" ?>
<undeployment
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java" >
  <service name="CarStore" />
</undeployment>
```

Die Deaktivierung sieht z.B. wie folgt aus:

```
java ... org.apache.axis.client.AdminClient CarStoreUnDeploy.wsdd
Processing file CarStoreUnDeploy.wsdd
<Admin>Done processing</Admin>
```

Ausblick

- Grid Services im Grid Computing

- Grid-WSDL (GWSDL)

© Universität Mannheim, Rechenzentrum, 2004.

Last modified: Sun Jul 4 14:14:48 CEST 2004

36. Web-CMS

- PHP-Nuke
- Zope
- Typo 3
- Midgard
- IONAS
- Framework der BWL

36.1. weitere Anwendungen

- Wikis
- BLOGs
- Syndication with RSS

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Wed Jul 7 22:39:26 CEST 2004

37. e-learning

- Einleitung
 - dotLRN
 - Demonstration
-

37.1. Einleitung

Was ist E-learning?

- Orts- und zeitunabhängiges Lernen
- Lerner zentriert und individualisiert
- Verschmelzung von Ausbildung und Internet

Technik:

- Internet (Web) als primärer Modus für Kommunikation und Präsentation
- Inhalte in Form von Texten, Bildern, Animationen, Audio, Video usw.
- Kommunikation wie Email, Chat, Bulletin-Board, Foren usw.
- Einsatz von Standardsoftware (Browser)

Inhalte:

- Interaktion mit Lehrern und anderen Studierenden
- *Geschlossene* Gruppenkommunikation
- Anpassung an individuellen Lernstil und -geschwindigkeit
- Verfolgung von Performance und Lernergebnissen

Plattformen:

- es gibt wohl über 600
 - Blackboard
 - Lotus Learning Space
 - Oracle iLearning
 - WebCT
 - Clix
 - Ilias
Open Source mit Unterstützung der Uni Köln
 - dotLRN
Open Source mit Unterstützung des MIT und der Uni Heidelberg
-

37.2. dotLRN

- www.mit.edu, MIT

- ocw.mit.edu, OpenCourseWare
- www.dotlrn.org, .LRN
- www.openacs.org, OpenACS

Technischer Aufbau

- dotLRN als Anwendung
- OpenACS als Middleware
- AOL/Netscape als Web-Server
- TCL/TK als Scriptsprache
- Postgres oder Oracle als SQL Datenbanken
- Unix (Linux, Solaris) als Betriebssystem

dotLRN ist eine Paketierung von OpenACS-Tools für eine Lernumgebung

dotLRn / OpenACS Module

- Datei-Verwaltung von Unterrichtsmaterialien
- Bulletin-Board System
- persönlicher und Gruppen Kalender
- News / Neuigkeiten
- Mailing Listen für Gruppen
- Hausaufgaben stellen und bearbeiten
- einfache Multiple-Choice Prüfungen
- komplexe Prüfungen
- Event Management für externe Interessenten
- Groupware für Forschungsgruppen
- Internationalisierung

Heidelberg und Mannheim

Ansprechpartner:

- Prof. M. [Hebgen](#), RZ Uni Heidelberg
- N. [Mazloumi](#), Wirtschaftsinformatik Uni Mannheim

Demonstration

- dotlrn.uni-mannheim.de, dotlrn@UniMa

Punkte:

- Übersichten auch ohne Einloggen
- Benutzerverwaltung wie rumms
- viele Installationen
- Eigene Startseite
- Eigener Kalender
- Mitgliedschaften in Veranstaltungen und Gruppen

Erstellt unter Verwendung von Vortragsunterlagen von M. Hebgen und N. Mazloui.

© Universität Mannheim, Rechenzentrum, 2004.

Last modified: Sat May 22 12:42:32 CEST 2004

38. Audio & Video

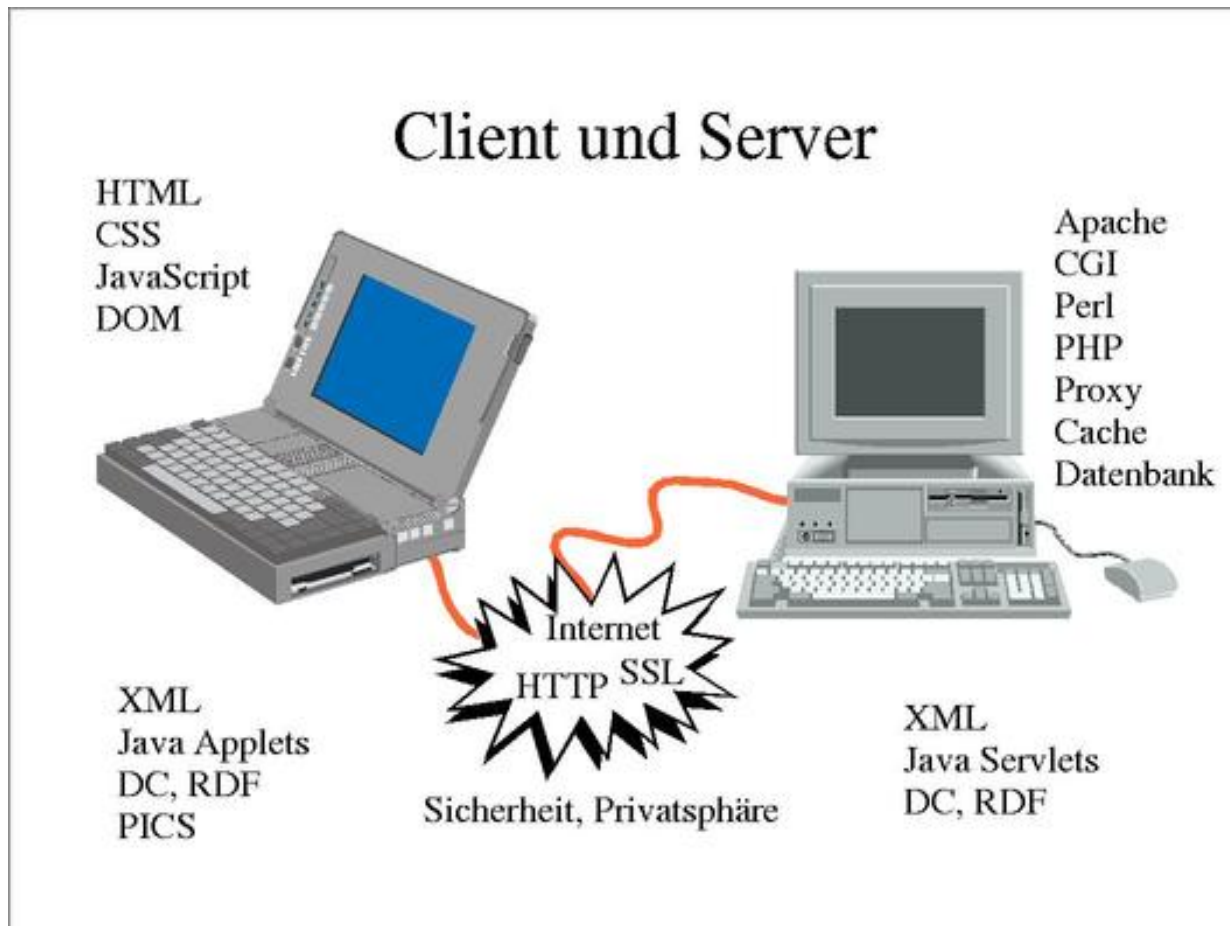
- Kompressionsverfahren
- Transport Protokolle
- RealPlayer
- SMIL, Synchronized Multimedia Integration Language

© Universität Mannheim, Rechenzentrum, 2002-2004.

Last modified: Sat May 22 12:42:43 CEST 2004

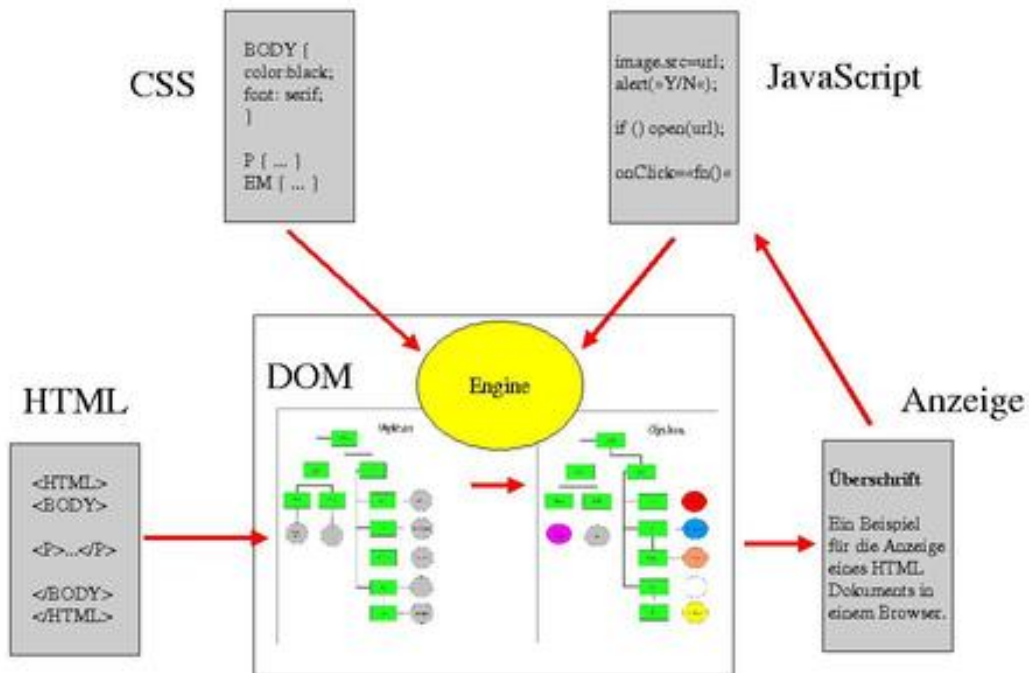
39. Schlussbemerkungen

39.1. Client - Server

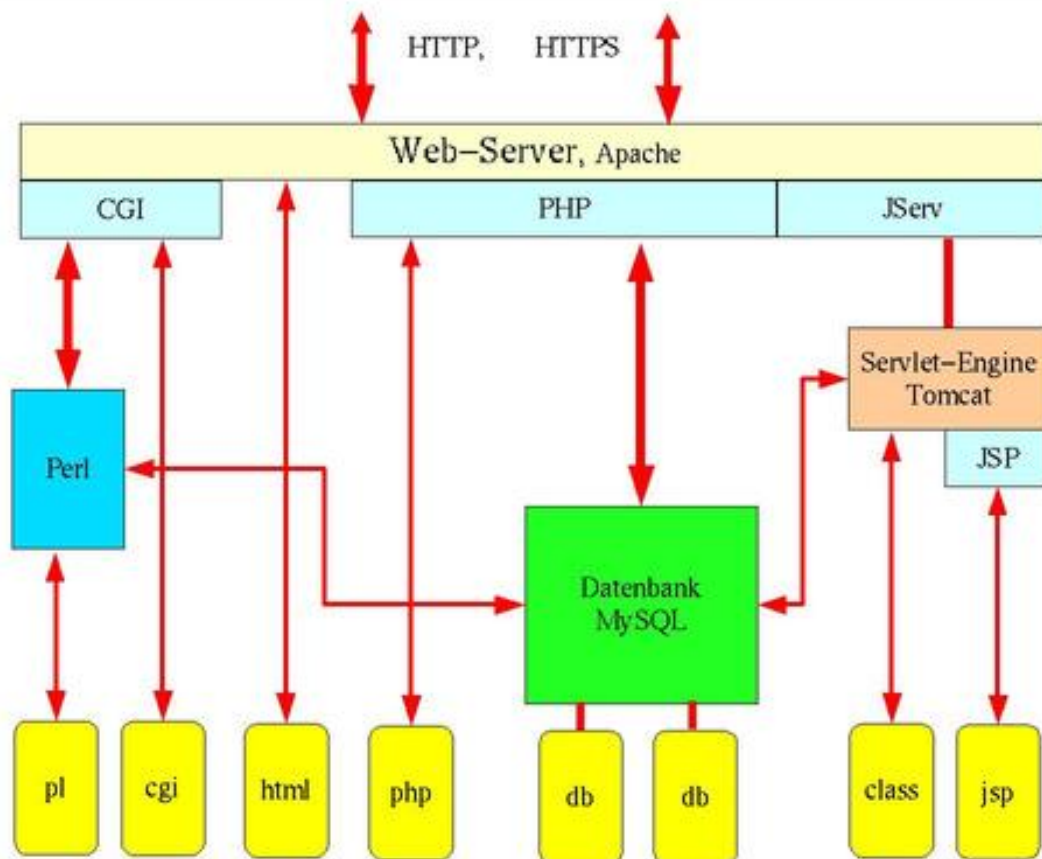


39.2. User Agents

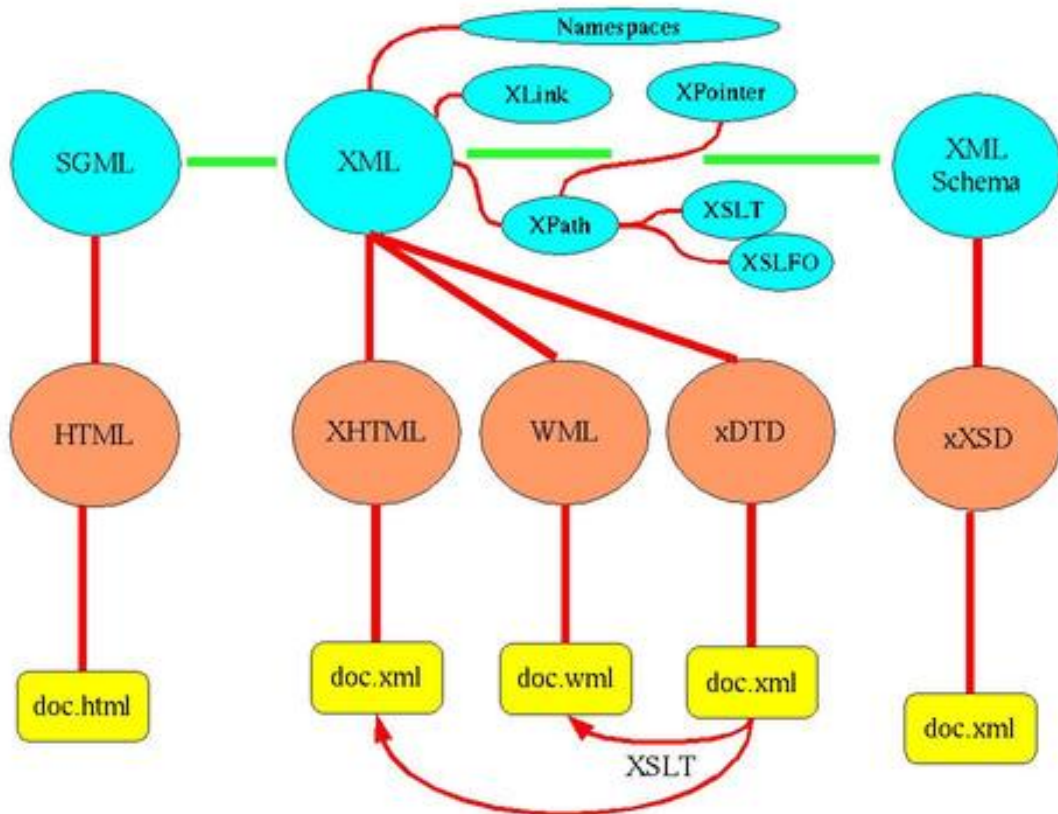
HTML, CSS, JavaScript und DOM



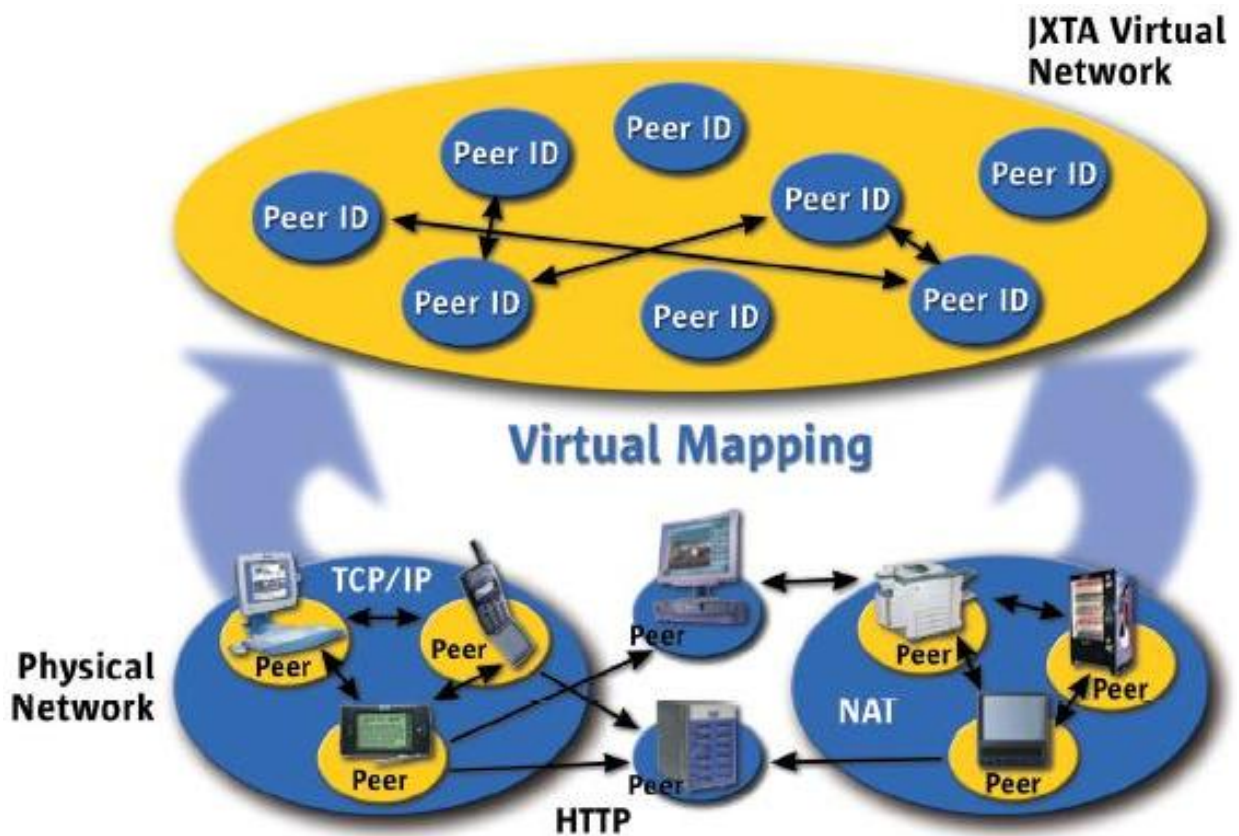
39.3. Server Architekturen



39.4. XML-Zoo



39.5. Peer to Peer



© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Sat May 22 12:51:36 CEST 2004

40. Literatur

Apache Web-Server

Apache Software Foundation, URL <http://www.apache.org/>

Apache. Das umfassende Referenzwerk

B. Laurie & P. Laurie, O'Reilly 1999.

Cascading Style Sheets

H. Lie & B. Bos, Addison-Wesley, 1997.

Cascading Style Sheets

S. Münz & A. Kefßler, Data Becker, 2001.

Cascading Style Sheets, level 1 (CSS1) Specification

W3C Recommendation, 1996, URL <http://www.w3.org/TR/REC-CSS1>

Cascading Style Sheets, level 2 (CSS2) Specification

W3C Recommendation, 1998, URL <http://www.w3.org/TR/REC-CSS2>

Computernetzwerke

A. Tanenbaum, Prentice Hall, 1996.

The Common Object Request Broker: Architecture and Specification.

OMG (Object Management Group), 1998, URL <http://www.omg.org/corba/corbiop.htm>

Content Management mit XML. Grundlagen und Anwendungen

G. Rothfuss & Ch. Ried (Herausgeber), Springer, 2002.

Cygwin = GNU + Cygnus + Windows

Portierung der GNU Tools auf Windows, URL <http://www.cygwin.com/>

Cygwin: A Free Win32 Porting Layer for UNIX® Applications

USENIX Windows NT Workshop Proceedings, 1998, URL <http://www.cygwin.com/usenix-98/cygwin.html>

Designing with JavaScript

N. Heinle, O'Reilly 1997.

Document Object Model (DOM) Level 1 Specification, Version 1.0

W3C Recommendation, 1998, URL <http://www.w3.org/TR/REC-DOM-Level-1/>

Document Object Model (DOM) Level 2 Specification, Version 1.0

W3C Working Draft, 1999, URL <http://www.w3.org/TR/WD-DOM-Level-2/>

dotLRN

URL <http://www.dotlrn.org/>

Dublin Core Metadata Element Set

Dublin Core Metadata Initiative, 1998, URL <http://purl.org/dc/>

Dublin Core Usage Guide

Dublin Core Metadata Initiative, 2000, URL <http://purl.org/dc/documents/wd/usageguide>

ECMAScript Specification

ECMA (European Computer Manufacturers Association), 1998, URL <http://www.ecma.ch/stand/ECMA-262.htm>

ECMAScript Components Specification

ECMA (European Computer Manufacturers Association), 1999, URL <http://www.ecma.ch/stand/ECMA-290.htm>

Extensible Markup Language (XML) 1.0 Specification

W3C Recommendation, 1998, URL <http://www.w3.org/TR/REC-xml>

Extensible Markup Language (XML) 1.1 Specification

W3C Recommendation, 2004, URL <http://www.w3.org/TR/xml11>

Extensible Style Language (XSL) 1.0

W3C Recommendation, 2001, URL <http://www.w3.org/TR/REC-xsl>

Gene Markup Language

URL (alt) <http://www.geml.org/>

Gene Expression Markup Language

URL (neu) <http://xml.coverpages.org/geneXML.html>

Groupware

C. Burger, dpunkt, 1997.

HTML 4.0 Specification

W3C Recommendation, 1998, URL <http://www.w3.org/TR/REC-html40/>

HTML und GCI Programmierung

R. Maurer, dpunkt, 1996.

HTML Ratgeber

H. Bonin, Hanser, 1996.

HTML Tidy

D. Raggett, W3C, 1999, URL <http://www.w3.org/People/Raggett/tidy/>.

Internet

RRZN, Uni Hannover, 1998.

Internet Engineering Task Force, IETF

URL <http://www.ietf.org/>.

Java

Sun, 1998-2000, URL <http://www.javasoft.com/> oder <http://java.sun.com/>

Java Apache

Apache Software Foundation, URL <http://java.apache.org/>

Java and the Java Virtual machine - Definition, Verification, Validation

R. Stärk & J. Schmid & E. Börger, Springer, 2001.

Java in a Nutshell

D. Flanagan, O'Reilly, 1997 (2nd ed.).

Java Programmierhandbuch und Referenz

S. Middendorf & R. Singer, dpunkt, 1999 (2nd ed.), 2002 (3rd ed.).

JavaScript

S. Koch, dpunkt, 1997.

JavaScript

D. Flanagan, O'Reilly, 1997.

JavaScript Developers Connection

Netscape, 1998, URL <http://www.developer.com/directories/pages/dir.javascript.html>

JavaScript Debugger

Netscape, 1998, URL <http://developer.netscape.com/software/jsdebug.html>

Java Server und Servlets

P. Roßbach & H. Schreiber, Addison-Wesley, 1999.

Java Server Pages

Sun, 2000, URL <http://java.sun.com/products/jsp/tomcat/> oder Apache Software Foundation, 2000, URL <http://jakarta.apache.org/>

Java Server Pages

H. Bergsten, O'Reilly, 2000.

Java Servlets

Sun, 2000, URL <http://java.sun.com/products/servlets/>

Java Software Engineering unter Linux

Oliver Böhm, SuSE PRESS, 2002.

Java und XML

B. McLaughlin, O'Reilly, 2000.

JXTA

Sun & JXTA.org, 2001, URL <http://www.jxta.org/> pzw. <http://www.sun.com/jxta/>

Koala, XSL Processor

INRIA, 1998, URL <http://www.inria.fr/koala/>

Learning Perl

R. Schwartz, O'Reilly 1993.

Lynda, Web-Design

URL <http://www.lynda.com/>

Microsoft

URL <http://www.microsoft.com/>

MySQL Database Engine

URL <http://www.tcx.se/>

MySQL and mSQL in a Nutshell

R. Yarger & G. Reese & T. King, O'Reilly 1999.

Netscape

URL <http://www.netscape.com/>

Online-Tutorial zu HTML (mit CSS)

S. Münz, 1998-1999, URL <http://www.teamone.de/selfhtml/>

OWL Web Ontology Language Guide

W3C Recommendation, Feb. 2004, URL <http://www.w3.org/TR/owl-guide/>

OpenACS

URL <http://www.openacs.org/>

OpenP2P

O'Reilly P2P, URL <http://www.openp2p.com/>

OpenSSL

URL <http://www.openssl.org/>

Opera Web-Browser

1999, URL <http://www.opera.com/>

PHP - Dynamische Webauftritte professionell realisieren

E. Schmid & Ch. Cartus & R. Blume, Markt & Technik 1999.

PHP - Grundlagen und Lösungen

J. Krause, Hanser 2000.

PHP: Hypertext Preprocessor

PHP Development Team, 1998, URL <http://www.php3.org/>

PHP Introduction

devshed, 1998, URL <http://www.devshed.com/resource/advanced/php3/intro/>

Platform for Privacy Preferences Project (P3P)

W3C, 1999, URL <http://www.w3.org/P3P/>

Practical Transformation Using XSLT and XPath

Ninth Edition, 2001-01-19, Crane Softwrights Ltd., URL <http://www.cranesoftwrights.com/training/>

Professional XML

R. Anderson & M. Birbeck & M. Kay & S. Livingstone & B. Loesgen & D. Martin & S. Mohr & N. Ozu & B. Peat & J. Pinnock & P. Stark & K. Williams, Wrox, 2000.

Programming Perl

L. Wall & R. Schwartz, O'Reilly 1991.

Programming the World Wide Web

R. Sebesta, Addison Wesley, 2003.

Protocol Extension Protocol (PEP)

W3C, 1998, URL <http://www.w3.org/Protocols/PEP/>

Resource Description Framework (RDF) Model and Syntax Specification

W3C Recommendation, Feb. 1999, URL <http://www.w3.org/TR/REC-rdf-syntax>

RDF Vocabulary Description Language 1.0: RDF Schema

W3C Recommendation, Feb. 2004, URL <http://www.w3.org/TR/rdf-schema/>

Request for Comment (RFC)

Online Verzeichnis, 1999, URL <http://www.rfc-editor.org/>

Rich in Style

Tipps zu CSS, 2002, URL <http://www.richinstyle.com/>

ServerWatch

Vergleich von Web- und anderen Servern, URL <http://www.serverwatch.com/>

Style-Guide für Web-Sites

1998, URL <http://www.gui-design.de/>

Unicode

URL <http://www.unicode.org/>

Die Sprache des Web: HTML 3, HTML 4

R. Tolksdorf, dpunkt, 1995, 1997.

Webreview: Vergleiche der CSS Unterstützung verschiedener Browser

URL <http://webreview.com/style/index.shtml>

Web-Server Funktionsvergleiche

URL <http://webcompare.internet.com/>

World Wide Web Consortium

W3C Homepage, URL <http://www.w3.org/>

Web Content Management - Websites professionell planen und betreiben

O. Zschau & D. Traub & R. Zahradka, Galileo Press, 2001.

Web-Technologien

C. Strobel, Oldenburg, 2004.

Web-Technologien, Konzepte - Programmiermodelle - Architekturen

H. Wöhr, dpunkt, 2004.

W3C Style Sheet Beispiele

W3C, 1997, URL <http://www.w3.org/Style/>, siehe auch <http://www.w3.org/StyleSheets/Core/preview>

Webmaster in a Nutshell

S. Spainhour & V. Quercia, O'Reilly, 1996.

Webmasters Handbuch

C. Neuss & J. Vromans, Thomson, 1996.

Webserver betreiben - HTTP und Apache: Grundlagen, Konzepte, Lösungen

J. Schröder & M. Müller, dpunkt, 1999.

The Wiki Way

B. Leuf & W. Cunningham, Addison-Wesley, 2001.

WWW

Ch. Meinel & H. Sack, Springer, 2004.

Wilde's WWW

E. Wilde, Springer, 1999.

Xitami Web-Server für Windows

1999, URL <http://www.imatics.com/>

XHTML 1.0: The Extensible HyperText Markup Language - A Reformulation of HTML 4 in XML 1.0

W3C Recommendation, January 2000, URL <http://www.w3.org/TR/xhtml1>

XML Apache Tools

Apache Software Foundation, URL <http://xml.apache.org/>

XML Handbook (5th ed)

Ch. Goldfarb, Prentice Hall, 2004.

XML Kompakt

Th. Michel, Hanser, 1999.

XML Resources

URL <http://www.xml.com/>

XML Resources at IBM

URL <http://www.ibm.com/xml/>

XSLT Programmers Reference

M. Kay, Wrox, 2000.

XSL-FO in der Praxis

M. Montero Pineda & M. Krüger, dpunkt, 2004.

XSL-FO Formater: AXF

Antenna House Inc. <http://www.antennahouse.com/>

XSL-FO Formater: FOP

Apache Software Foundation, URL <http://xml.apache.org/fop>

© Universität Mannheim, Rechenzentrum, 1998-2004.

Last modified: Thu Jul 15 22:14:21 CEST 2004