1. Die Basis: Netze und Protokolle

1.1 Geschichte

- „Internet“ seit ~ 1969, 1982 TCP/IP, 1997 B-WIN, heute X-WIN
- das World Wide Web (Hypertextsystem, -links) seit 1989 (Tim Berners-Lee am CERN)

1.2 Grundbegriffe

- DTE: Datenendeinrichtung – Eingabe, Ausgabe, Speicher, …
- DCE: Datenübertragungseinrichtung – Rechnersystem für Auf-/Abbau von Datenverbindungen, Erkennen von Übertragungsfehlern, …
- Datenübertragung: Kommunikation zwischen Computern
- verbundene Rechner bilden Rechnernetz
- NAT: Network Address Translation
- DNS: Domain Name Service

100m	Local Area Network: LAN	Gebäude, Campus
100m	Local Area Network: LAN	Gebäude, Campus
10 km	Metropolitan Area Network: MAN	Stadt
1.000 km	Wide Area Network: WAN	Land
10.000 km	Internet (Global Area Network: GAN)	die Erde – und mehr?

1.3 Kommunikationssysteme

- direkte Vernetzung: DTEs direkt verbunden
- VPN: virtuelles privates Netzwerk
- erste Rechnernetze: Point-to-Point
- Switching Networks: alle über eine Leitung
- Prinzip der Paketvermittlung: zerlegt eine Nachricht in einzelne Pakete, dieses besteht aus Empfänger, Sender, Sequenznummer, Prüfsumme und natürlich den Daten
- **verbindungslose Netzwerkdienste** (Datagramm-Netzwerke): jedes Paket nimmt indiv. Weg
- **verbindungsorientierte Netzwerke**: Aufbau einer virtuellen Verbindung, Pakete nehmen selben Weg, uneffizient aber Dienstgarantie (früher Telefonie)
- Kenngrößen von Netzwerken: Geschwindigkeit, Korrektheit, Zuverlässigkeit, Verzögerung

1.4 Schichtenmodell

	Bedeutung	Bereich	Beschreibung	Beispiele
1	bit-Übertragung	Transport	Kabel, Steckerverbindungen (Gerät: HUB)	ITU-T V.24
2	Sicherungsschicht	Transport	zuverlässige Verbindung bei PtP (Gerät: Switch)	Ethernet
3	Vermittlungsschicht	Transport	Zuweisung von Adressen (Gerät: Router)	ICMP
4	Transportschicht	Transport	stellt zuverlässige Datenübertragung her	TCP/UDP
5	Sitzungsschicht	Anwendung		
6	Darstellungsschicht	Anwendung		
7	Anwendungsschicht	Anwendung		

1.5 IP-Adressen

Befehle:
nslookup ADRESSE
telnet SERVER PORT
HEAD HTTP/1.1
ftp (Port 21)
ssh (Port 22)

- Class-C: die letzten 8 bit frei: 256 Adressen
- Class-B: die letzten 16 bit frei: 65.636 Adressen
- Class-A: die letzten 24 bit frei: 16.777.216 Adressen
- private Adressen: 10.x.x.x, 172.16.x.x -172.31.x.x, 192.168.x.x
- TCP: zur Adresse kommt noch der Port (16 bit)
- SMTP: Simple Mail Transfer Protocol
- POP: Post Office Protocol
- IMAP: Internet Message Access Protocol

2. & 3. Der Apache Web Server Teil I & II

2.1 Bedeutung und Geschichte des Apache

- der Standard wird der gepachte NCSA-Server: „a patchy server“ = Apache (Legende)
- Server-Anwendung auf TCP-Port 80, versteht http
- 1995 Beta des Apache-Webserver veröffentlicht, heute (Okt 2011) 2.2.21
- ab Apache 2.0 IP v6 Unterstützung

```
<apachedir>/bin/apachectl start
```

```
./configure                          \
--prefix=/home/zxmxm48/apache2       \
--with-port=33581                    \
--enable-info                        \
--enable-ssl                         \
--with-sslport=33582                 \
--enable-rewrite
make && make install
```

2.2 Installation

2.3 Prinzip der Konfiguration

- Einstellungen nach der Installation werden in httpd.conf vorgenommen
  Teil I: Allgemeines (Servername, Port, …)
  Teil II: Direktiven des Hauptservers (Regeln für einzelne Verzeichnisse)
  Teil III: Virtuelle Hosts (nicht behandelt)
- Paramenter sind case-sensitiv
- Direktiven: absolute Pfade, kein abschließender „/“
  Location: geht vom DocRoot aus
- Includes: Direktive für ServerSideIncludes

```
<Directory />
    Options None
    AllowOverride None
    Deny from all
</Directroy>
<Directory „/home/zxmxm48/public“>
    Allow Override All
    Oder allow,deny
    Options All
    Allow from all
</Directory>
```

2.4 Sonstiges

- Analyse von Apache-Logs (webalizer, awstats, Google Analytics)

3. Module

- je weniger Module desto schneller und sicherer
- **statisch**: in den Apache Kern hineincompiliert und sind immer verfügbar
- **dynamisch**: werden beim Starten eingebunden (Drittmodule sind dynamisch), werden über die Direktive LoadModule in httpd.conf geladen

3.1 Module der ASF

- mod_core: Kerndirektiven wie Directory und DocumentRoot (muss vorhanden sein)
- mod_access: Zugriffsteuerung auf Apache über Hostnamen oder IP-Adressen
- mod_actions: Zuordnung von MIME-Typen zu CGI-Scripten
- mod_auth: passwortgeschützte Inhalte (einfachste Variante der Authentifizierung)
- mod_autoindex: wenn keine indexdatei vorhanden, wird ein Verzeichnisindex erzeugt
- mod_cgi: Ausführen von CGI-Scripten
- mod_dir: legt Standards bei Zugriff auf Verzeichnis fest (DirectoryIndex index.html …)
- mod_env: Zugriff auf Umgebungsvariablen für Apache bzw. Script
- mod_include: Server Side Includes (SSI)
- mod_so: ermöglicht dynamische Module (dieses muss aber statisch geladen warden)

3.2 Drittmodule

- mod_perl: integriert Perl-Interpreter (ebenso mod_python, mod_ruby, …)
- mod_php: PHP-Funktionalität
- mod_rewrite: Umschreiben der URL durch regulären Ausdruck
- mod_speling: korrigiert Schreibfehler in der URL
- mod_ssl: Schnittstelle von Apache zu OpenSSL

4. fastCGI und ServerSideIncludes

4.1 FastCGI

- Ansatz zum wesentlichen Beschleunigen von Server-Anwendungen (nicht selbst durchgesetzt, aber Basis für ähnliche Techniken (z.B. Ruby on Rails))
- CGI Serverstruktur: Web-Client, Webserver, (opt. Datenbankserver)
- Server legt beim Aufruf Umgebungsvariablen fest die im CGI-Programm verwendbar sind (z.B. SERVER_NAME, SERVER_PORT, REMOTE_HOST, REQUEST_METHOD, …)
- Vorteile CGI: einfach, Sprachunabhängig, offener Standard, Architekturunabhängigkeit
  Nachteile: stets neuer Prozess → keine optimale Performance
- Vorteile fastCGI: Performance + CGI-Vorteile
- fastCGI-Prozesse werden nach einer Anfrage nicht beendet sondern warten auf die nächste
- Webserver und fastCGI-Server können getrennt werden (kommunizieren über TCP)
- Ablauf: Webserver baut bei Anfrage die Verbindung zu fastCGI-Prozess auf und sendet darüber die CGI-Umgebungsvariablen und stdin; fastCGI sendet stdout und stderr zurück
- Nutzung erfordert den Apache-Mod mod_fastcgi

4.2 ServerSideIncludes (SSI)

- simpelste Möglichkeit für dynamische Seiten, vollständig serverbasiert
- benötigt Apache-Modul mod_include (statisch oder dynamisch integrierbar)
  - Options +Includes (eingeschränkt mit IncludesNOEXEC)
  - AddType text/html .shtml, AddHandler server-parsed .shtml und ausführbar sein
- SSI sind HTML-Anweisungen die serverseitig bei Auslieferung der Seite ausgewertet werden (z.B. Last-Modified des Dokuments, einbinden einer Datei)
- SSI Commands: include, config, echo, fsize, flastmod, exec
  Beispiel: <!--#echo var="SERVER_NAME" --> <!--#include virtual="/footer.html" --> <!--#exec cmd="ls –alp" --> (führt ein Befehl aus!)

5. Content-Management-Systeme (Typo3)

- vollständiges Benutzer-/Rollenkonzept, Content mit Zeitbeschränkung, Mehrsprachigkeit
- dynamisches CMS: erzeugt Seite bei jedem Aufruf neu, immer aktuell, langsamer
  statisches CMS: erzeugt statischen Abzug im Filesystem, performanter, stabiler
  hybrides CMS: Mischform der oberen beiden
- Typo3 seit 1997: serverseitig PHP-Code, MySQL, Backend: Admin, Frontend: Website
- typo3conf/localconf.php stehen die meisten Optionen
- grundlegende Komponenten:
  - CSS für die grundlegende Formatierung
  - TYPO3-Template mit TypoScript
  - HTML-Template zur Verbindung des Inhalts mit dem Typo3-Template
- Extensions: realurl (Umsetzung der URL von index.php?id=xx in was leserliches)
- Nachteile: sehr komplex, Extensions sein teilweise instabil, hohe Anforderungen an die Hardware, Security nicht unproblematisch, TypoScript unbeliebt

6. Architekturen von Web-Applikationen

- Die Architektur der Web-Anwendung webstimmt maßgeblich deren Qualität
- Architektur für die Anbindung des Applikations-Servers an den Web-Server und für die Integration bestehender Anwendungen und Datenbestände
- Web-Server nimmt HTTP-Requests entgegen und leitet diese an den Application Server weiter
- Verwendung eines Webservers bietet Stabilität, Security, Performance, aber höhere Kosten
- übliche Forderungen an die Architektur: Wirtschaftlichkeit, Kongruenz der Lösungen in einer Firma, Einbindung bestehender Applikationen
- die Architektur beschreibt Struktur, macht Systeme verständlich, ist der Rahmen für flexible Systeme und ist der Übergang zwischen Analyse und Realisierung
- Kategorisierung von Architekturen:
    - Schichtenaspekt: Trennung in Schichten (tiers) zur Trennung von Verantwortlichk.
      2-tiers: client-Schicht, Application-Server mit Datenbank
      3-tiers: Web-Präsentation, Application-Server, Datenhaltung
      n-tiers: Client, Firewall, Proxy, Web-Server, Application-Server, Datenbank, …
    - Datenaspekt: strukturierte Daten VS. nichtstrukturierte Daten
- **Proxy**: Proxy zur Performance-Optimierung (Caching-Proxy), Link-Proxy, History-Proxy
- Problem bei größeren multimedia Daten wird alternative zu HTTP benötigt
  Lösung: Live-Stream, Video on Demand (dazu gesicherte Bandbreite notwendig)

7. Template-Engines am Beispiel Smarty

- **Template Engine**: Software, welche in Vorlagen Platzhalter mit konkretem Inhalt ausfüllt, in Templates soll keine Business-Logik enthalten sein, Trennung von View und Controller
- **Bewertung**: Designer arbeitet unabhängig vom Programmierer
- Smarty ist eine kompilierende Template-Engine für PHP
  Vorteile: sehr performant, nur einmaliges Compilieren der Templates, Caching, Erweiterbar
- Caching: fetch(), display()
- Smarty 3: neuer Template-Parser, Vererbung von Templates, Funktionsdefinition und –aufruf im Template, …

```
{* Kommentar *}
{$ARRAY.key}
{$REFERENZ->attribut}
{$smarty.server.SERVER_NAME} …
{foreach from=$ARR item=VL}
{include file="footer.tpl"}
{insert file="footer.tpl"}
{assign var="name" value="value"}
{fetch file="datei"}
```

8. ZEND-Framework

- **Framework**: wiederverwertbares Softwaresystem mit bereits implementierter, generierter Funkt. setzt Architektur um und folgt Design Pattern
- Frameworks mindern die Performance, da deutlicher Overhead (Optimierung sinnvoll durch Caching des PHP-Codes)

8.1 ZEND

- MVC direkt abgebildet, sehr flexible Lizenz, auch kommerzieller Einsatz (ZEND-Studio)
- Installation: wie Smarty in php.ini den include_path anpassen
- mittels Rewrite werden alle Requests an die zentrale index.php weitergeleitet, die dann entsprechend umverteilt
- Benutzer und Rechtemanagement mit Klasse Zend_Acl
- Zend_Session, Zend_Validate, Zend_Filter, Zend_Config

```
require_once(‚Zend/Loader.php‘);
Zend_Loader::loadClass(‘Zend_Beispiel‘);
```

```
zf.sh ?
zf.sh create project <myzf> (Projekt erstellen)
zf.sh create controller <Name>
zf.sh create action <Name> <Controller>
```

- weitere Möglichkeiten: Web-Services, Captcha, pdf-Generierung, Mail, etc.
- Vorteil: keine eigene Templatesprache sondern reines PHP

9. MVC-Frameworks: Ruby on Rails & CakePHP

9.1 Ruby

- moderner Nachfolger von Perl, sauber objektorientiert
- implementiert MVC-Paradigma, Konvention ersetzt aufwendige Konfiguration
- Fünf Module:
  - M: Active Record
  - V: Action Pack (Request-/Reponse-Behandlung)
  - Action-Mailer (Email)
  - Action Web-Service
  - Active Support (Ruby-Erweiterungen)
- **Scaffolding** erlaubt die Entwicklung von Web-Applikationen „On the fly“
- Create, Read, Update, Delete aus Dankenbank wird mit der Anwendung erzeugt
- **Model**: eines der unterstützten DBMS
  Generator erzeugt Skeleton einer Ruby-Klasse
- **Controller**: Instanz der ActionController-Klasse verarbeitet Request
  Std-C für Zugriff: new, list, delete, update, show
- **View**: Klasse ActionView, Template enthält im Kern HTML, bereichert um Ruby-Elemente
- erzeugen einer Rails-Applikation: rails <path_to_appl> (erzeugt entsprechende Unterordner)
- Kurzfassung: bietet ein „fast-development“ → Prototyping, direkte Anpassung an das Datenmodell in DB, MVC automatisch und zwingend, mit Template-Engine, automatisches Mapping auf Controllermethoden, minimale Konfiguration

```
ruby script/generate model <Model>
ruby script/generate controller <Name>
ruby script/generate scaffold <Model> (C & M zusammen)
```

```
def list
    … code …
end
```

```
<% ruby %>
<%= rubyVariable %>
```

9.2 CakePHP

- Grundprinzipien
  - DRY: don’t repeat yourself
  - Rapid Development: kurze Planungsphase, schnell ausführbarer Code
  - CRUD: direkte erstellung dieser vier DB-Operationen
  - Convention over Configuration
- Installation: Apache-Direktive AllowOverride all für .htaccess-Dateien
- Cake-Konsole zum generieren von Code
- $scaffold aktiviert Grundgerüst mit Methoden, index, view, add, edit, delete
- Konfiguration: app/config/core.php, database.php, routes.php
- URL: http://domaine/controller/action/paramater1/parameter2
- Konventionen: Dateinamen: Kleinbuchstaben und _
  gleiche Bezeichnung für Model und DB-Tabelle im Plural (students ⇔ Student)
- View in HTML mit PHP-Code, Helper-Klassen zur Vermeidung von doppeltem Code

9.3 Web 2.0

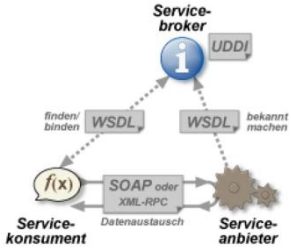
- Kern-Idee: Web 1.0 ist veröffentlichen, Web 2.0 ist teilnehmen (typisch: Ajax, Rails, …)

10. Web-Services

- ein Webservice ist ein Service, der von anderen Softwaresystemen über ein Netzwerk und möglicherweise auch dem Internet über SOAP aufrufbar ist.
- Maschine-Maschine-Kommunikation (Unterschied zum klassischen Web)
- Web-Services sind für die maschinelle Nutzung, sie bieten keine Benutzerschnittstellen
- Adressierung über Uniform Ressource Identifier (URI), Schnittstellen XML, Internet-Protokolle (z.B. Google-Suche, Shop-Suche)

10.1 UDDI (Universal Description, Discovery and Integration)

- Verzeichnisdienst zum Registrieren von Web-Services
- *Vorteile*: offene Standards, keine Lizenzkosten, offene und flexible Architektur, einfacher Einstieg, verschiedene Protokolle, wie HTTP, Java-RMI, COBRA, …
  Nachteile: Sicherheit (Teillösungen: SSL, VPN, Firewalls), Performance, einzelne Techniken sind komplex



10.2 XML-RPC

- Prinzip: HTTP-POST-Request, dieser enthält Methodenaufruf und übergibt Parameter Antwort in einem der 8 XML-RPC-Typen in XML-Struktur
- verwendet gängige Datentypen: Integer (32), Double (64), String, Boolean, dateTime, …

10.3 SOAP (Simple Object Access Protocol)

- … ist weder simple noch dient es dem Objektzugriff
- flexibler als XML-RPC: auch asynchrone Kommunikation, Peer-to-Peer Kommunikation

10.4 Beschreibung von Webservices

- **WSDL** (Web Services Description Language): Sprache zur Beschreibung der unterstützten Methoden und ihrer Parameter
- XML-Dokument, das den Dienst korrekt beschreibt (definitions, types, message, protType, …)
- Standards für Sicherheit: XML-Signature, XML-Encryption, SAML (Sec. Assert. Markup Language)

11. Security

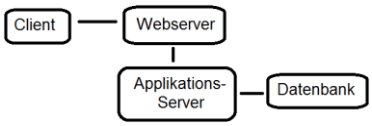
- **Datenschutz**: Schutz des Einzelnen vor Missbrauch der eigenen Daten (data privacy)
- **Datensicherheit**: Vertraulichkeit, Verfügbarkeit und Integrität intformationsverarbeitender Systeme

11.1 Kurze Theorie

- **Exploit**: Aufzeigen einer Lücke durch Beispielschadprogramm oder dessen Beschreibung
- **IDS** (Intrusion Detection System): erkennen der Korrumtion eines Systems (Snort)

11.2 Die Situation

- der Webserver muss weltweit erreichbar sein und HTTP-Requests bedienen
- manche Software installiert Webserver versteckt mit, diese sind häufig veraltet und damit anfällig



11.3 Das Netzwerk & der Host

- diejenigen, die das Netzwerk konfigurieren, können bereits sehr früh zentrale Vorkehrungen treffen
- nur Webserver muss von außen erreichbar sein (80, 443, 22) alle anderen Ports blockieren
- nur notwendige Dienste betreiben, Standard-Ports ändern (z.B. Datenbank-Port)
- Verteilung der Rechner auf verschiedene Netze

11.5 Der Webserver

- Benutzerkennung für Apache-Dienst (niemals root), Kennwortdateien über .htaccess auslagern
- Zugriffsrechte auf Verzeichnisse, Systemdateien nicht public speichern
- CGI separat von htdocs, Verzeichnis nicht lesend und für Suchmaschinen verbieten
- unverschlüsselte Protokolle vermeiden (kein FTP, kein telnet), SSH nur mit Zertifikat
- Suchmaschinen verbieten alles zu durchsuchen, Serverinfos sperren (server_info, _status)
- DOS (Denial of Service)-Angriffe abwehren durch Änderung der IP-Adresse, …

11.6 Die anderen Server

- Appikation-Server sind besonders anfällig (CIG, PHP, J2EE, Wordpress, Forensoftware wie phpBB)
- nur die notwendigen Dienste betreiben, updaten und Beispielscripte entfernen
- phpinfo() nicht für alle veröffentlichen, konfiguration in der php.ini (register-globals = off)

11.7 Informationen

- ein echter Webserver kann nicht so mal nebenbei betrieben werden (Kontrolle, Updaten)
- Informieren über aktuelle Sicherheitslücken und seinen eigenen Server angreifen (SATAN)
- sichere Passwörter, sichere Protokolle (SSH), eigene Backups aller Daten!
- SQL Injection, Same Origin Policy, Session Hijacking, ARP/DNS/URL Spoofing, Ajax Security, Flash Security, Buffer Overflow, URL erraten (von Downloads)

12. Rechtliche Aspekte

Teledienste und Mediendienste

- Teledienste: elektronische Inforamtions- und Kommunikaitonsdienste (z.B. Homebanking)
- Mediendienste: elektronische Verteildienste und soclhe, bei denen die redaktionelle Gestaltung zur Meinungsbildung im Vordergrund steht (z.B. Tageszeitung im Web)
- einheitliche wirtschaftliche Rahmenbedingungen für die verschiedenen Nutzungsmöglichkeiten der Informations- und Kommunikationsdienste, Schutz personenbezogener Daten bei Telediensten

Rechtliche Anforderungen

- Anbieterkennzeichnung durch Impressum: Verantwortliche, Firma, Anschrift, Hosting-Service
- mit Einwilligung des Nutzers ist zulässig: Erhebung der Emailadresse für Zusendung von Newslettern, Teilnehmerverzeichnis, Erhebung und Verarbeitung personenbezogener Bestandsdaten für Zwecke der Beratung, der Werbung, der Marktforschung
- unzulässig: pauschale Einwilligung in die Nutzung der erhobenen Daten für andere Zwecke in AGB, obligatorische Erhebung von Daten, die für die Erbringung des Dienstes nicht relevant sind
- Realisierung der Zustimmung durch eindeutigen Button oder Bestätigungsmail
- Nutzer hat jederzeit das Recht, eine Einwilligung zu widerrufen
- Auskunftsrecht über die Daten, die der Anbieter über ihn gespeichert hat
- Weiterleitung an Dritte ist eindeutig zu kennzeichnen

TKG: Telekommunikationsgesetze

- TKG: regelt Wettbewerb im Bereich der Telekommunikation, wesentlich verändert in Folge 9-11 in Erweiterungen (Vorratsdatenspeicherung, Kinderpornographie)

Landesdatenschutzgesetz LDSG

- den einzelnen zu schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seiner Persönlichkeit nicht beeinträchtigt wird: Selbstbestimmung/Recht an eigenen Daten