Anwendungsorientierte Informatik

Foliensammlung

Organisatorisches

- Termine
  - ✔ Beginn 5.10.2018
  - ✔ Jeweils Freitags 14:00-17:15 Uhr (V+Ü)
- Zeitplan/Folien/Unterlagen/Skripte
  - ✔ http://knopper.net/bw/ainf/
- Änderungen zu Veranstaltungen vor 2011:
  - ✔ Keine „Anwensenheits-Punkte“ (Klausur/Prüfung = 100%)
  - ✔ +Kurzeinführung „Betriebssysteme & Anwendungen“
  - ✔ Plattformunabhängige Übungen, Lösung immer unter verschiedenen Betriebssystemen (Windows, Linux, Mac) möglich.

Vorlesungsinhalte (1)

- Softwareentwicklung & -auswahl
  - ✔ 1-2 Vorlesungen "Allgemeines zu angewandter Informatik, Einführung in Betriebssysteme und Anwendungen allgemein + Beispiele Zusammenhang mit „Softwaretechnik für Nichtvertiefer“, Softwareentwicklung u. Übersicht Programmiersprachen, Lizenzen und juristische Fragen.
  - ✔ 2 Vorlesungen „3D-Konstruktion und 3D-Druck“ (Programmieren mit OpenSCAD)

Vorlesungsinhalte (2)

- Internet-orientierte Informatik
  - ✔ 1 Vorlesung "HTML Einführung" (inkl. Tabellen und Formulare),
  - ✔ 1 Vorlesung "Javascript" (Client-side Web-based Applications),
  - ✔ 1 Vorlesung Internet TCP/IP und Netzdienste
  - ✔ 2 Vorlesungen "PHP" (Server-side Web-based Applications).

Vorlesungsinhalte (3)

- Datenbankorientierte Informatik
  - ✔ 1 Vorlesung "SQL" (direkte Kommunikation mit der Datenbank per MySQL-Client),
  - ✔ 1 Vorlesung "PHP-MySQL" (Webformulare + Datenbankanbindung in HTML+PHP),
  - ✔ 1 Vorlesung "ERM" mit Diagrammen, Zusammenhang mit Web-Applikationen in verschiedenen Use Cases.

Zur Benutzung der Folien

- Zusammenfassung prüfungsrelevanter Themen
- Verweise auf (ebenfalls prüfungsrelevante) Handouts, die mehr ins Detail gehen
- Angabe von *zusätzlicher* Sekundärliteratur, teils Online-Artikel (☞ Links), *Lesen dringend empfohlen!*

Zu den Übungen

- Vertiefung der Theoriethemen durch praktische Anwendung
- Eigene Lösung wird zwar nicht direkt benotet, aber dringend empfohlen, da Inhalte auch *prüfungsrelevant*!

Betriebssysteme und Anwendungen (1)

Systemsoftware

Hierzu gehören das ☞ Betriebssystem (inkl. „Treiber“ bzw. „Kernel“) des Computers sowie viele Systemdienste und -programme, die die Hardware-Ressourcen des Computers erst benutzbar machen.
→ Übung: Betriebssystemstart mit KNOPPIX-DVD oder Stick
→ Handout „Bootvorgang“.

Anwendersoftware

Hierzu gehören die Programme, mit denen der Computer-Nutzer direkt arbeitet. Unter ☞ Unix zählt (im Gegensatz zu ☞ Windows) neben den Anwendungen (Office, Datenverarbeitende Programme, Spiele, Internet-Nutzungssoftware) auch der graphische ☞ Desktop zur Anwendersoftware und kann auch ausgetauscht werden.

Betriebssysteme und Anwendungen (2)

Kompatibilität

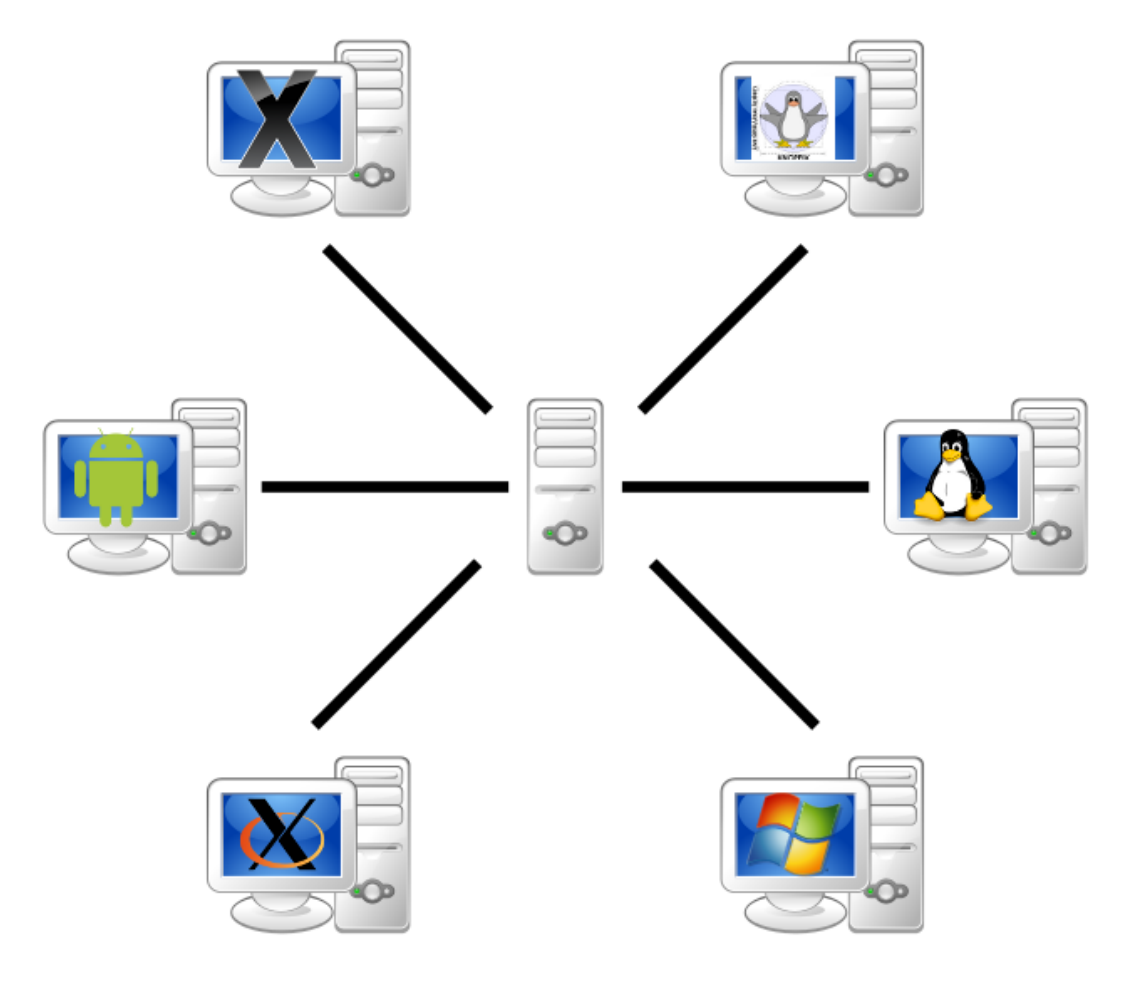
Jedes Betriebssystem stellt eine ☞ Laufzeitumgebung für Anwenderprogramme zur Verfügung. Aufgrund der unterschiedlichen Schnittstellen (☞ APIs) sind diese leider oft untereinander inkompatibel, d.h. beispielsweise:

- Windows-Programme laufen nicht unter ☞ Linux,
- Linux-Programme laufen nicht unter Windows,
- Programme für neuere Betriebssystem-Versionen laufen nicht mit älteren Versionen zusammen und umgekehrt,
- Programme für eine ☞ Prozessorarchitektur laufen nicht auf einer anderen.

Für diese Probleme mit den verschiedenen ☞ Systemen gibt es Lösungsansätze: ☞ Emulatoren, ☞ Virtualisierung, ☞ cross-plattform Anwendungen mit ☞ Client-/Server Architekturen.
→ Handout „Windows-Programme-unter-Linux...“

Stichwort Client-/Server Architektur

Wir merken uns diesen Begriff schon mal vor für die „internetorientierte Informatik“!



Systemorganisation: Dateisysteme

Um identifizierbare Objekte (Multimedia-Dateien, Dokumente, Programme, ...) auf Datenträgern schreiben und wieder von ihnen lesen zu können, werden diese bei fast allen Betriebssystemen in einem Ordnungssystem mit hierarchisch angeordneten Verzeichnissen abgelegt.

Unterschiede: Während Windows aus historischen Grunden allen Datenträgern „Laufwerksbuchstaben“, und jedem Laufwerk eine individuelle Verzeichnisstruktur gibt, ist unter Unix nur ein einziger Verzeichnisbaum vorhanden, beginnend mit dem Wurzelverzeichnis /, außerdem werden dort alle Datenträger, sogar „Netzlaufwerke“, in einen frei wählbaren Unterordner dieses Verzeichnisbaums „montiert“ (☞ mount).

→ Handout „Datentraegerverwaltung“

Softwarenutzung

- Programme zum Bearbeiten von Daten:
  - Formatierte oder „reine“ Texte (Textverarbeitung, Editor)
  - Grafik (Grafik-/Zeichenprogramm)
  - Durch Formeln oder Funktionen verknüpfte Daten (Tabellenkalkulation, Formulare)
- Programme zum Zugriff und Darstellung von Information (Browser, Viewer)
- Programme zum Erstellen von Programmen
  - ✔ Programmiersprachen mit
    - ✔ Compiler (Übersetzt Quelltext in Maschinencode)
    - ✔ Interpreter (führt Anweisungen direkt aus)
- Entwicklungsumgebungen
  - ✔ Editor mit Verknüpfungen zu Dokumentation und Hilfsfunktionen
- Graphische Oberflächen (GUIs)
- Spiele und Entertainment

Anwendersoftware, Multiplattform-Beispiele

- Textverarbeitung, Tabellenkalkulation, Präsentation, Zeichnungen, Datenbank-Anbindung: OpenOffice / ☞ LibreOffice
- Grafikbearbeitung: Gnu Image Manipulation Program (☞ GIMP)
- WWW: ☞ Apache WWW Server, ☞ Mozilla Firefox Browser
- E-Mail/Groupware: ☞ Evolution, ☞ Mozilla Thunderbird
- Multiprotokoll Chat: ☞ Pidgin
- Videokonferenz: ☞ Ekiga
- Datenbanken: ☞ MySQL

Software-Entwicklung

- Komplexe (umfangreiche) Software wird meist im Team entwickelt
- Um die Arbeit zu koordinieren und die Übersicht zu behalten, kommen verschiedene Vorgehensweisen und Methoden zum Einsatz
  ☞ Software Engineering
- ☞ Programmiersprachen erlauben es, ☞ Algorithmen mit hohem Abstraktionsgrad für den effizienten Ablauf im Computer zu realisieren. (NB: Ein Algorithmus ist allgemein eine Schritt-für-Schritt Anleitung zur Lösung eines Problems.)
- Neben Sprachen zum Programmieren gibt es standardisierte “Sprachen“ für verschiedene Dateiformate und ☞ Protokolle.

Rechtliche Aspekte von Software (1)

- Der Urheber der Software (Autor) legt die Nutzungsbedingungen für sein Programm fest, wobei er Rechte Dritter beachten muss.
- Proprietäre Software: Software wird „vermietet“, d.h. Nutzung gegen Gebühr, alle anderen Rechte verbleiben beim Rechteinhaber.
- Open Source/ Freie Software: Software wird als frei verfügbarer „Baukasten“ verteilt, die Empfänger erhalten weitgehendes Eigentumsrecht an der Software inkl. Weiterentwicklung und Verbreitung. Geld wird hier hier eher mit Service und Support verdient, nicht mit „Nutzungslizenzen“.
  ☞ http://www.opensource.org/
- Jede Lizenz ist ein Vertrag zwischen zwei Parteien, der akzeptiert werden kann, oder auch nicht.

Rechtliche Aspekte von Software (2)

Tabelle: Übersicht Lizenzmodelle und Rechte

	Nutzung kostenlos	Frei kopier bar	Zeitlich unbe grenzt nutzbar	Quelltext wird mitgelie fert	Bearbei tung erlaubt	Einbau in proprie täre Produkte erlaubt	Änderung der Lizenz für Bearbei tungen mögl.
Proprietäre Software							
Shareware	✔	✔					
Freeware	✔	✔	✔				
GPL	✔	✔	✔	✔	✔		
LGPL	✔	✔	✔	✔	✔	✔	
BSD	✔	✔	✔	✔	✔	✔	✔

Literatur zum Urheber- und Internetrecht

http://www.uni-muenster.de/Jura.itm/hoeren/lehre/materialien
Professor Dr. Thomas Hoeren, Institut für Informations-, Telekommunikations- und Medienrecht an der Universität Münster, Kompendium zum Internetrecht (PDF)

Bitcoin und Blockchain (Wunschthema?)

Kaum eine andere Technologie hat derzeit das Potenzial, betriebswirtschaftliche Anwendungen im Finanzwesen so zu revolutionieren wie die Blockchain, die sich quasi als „Nebenprodukt“ aus der ersten populären elektronischen Währung Bitcoin ergeben hat.

→ Paper von (Pseudonym) Satoshi Nakamoto

→ Separater Foliensatz „Bitcoin & Blockchain“

Beispiel OpenSCAD

S. Foliensatz „OpenScad-Tutorial.pdf“,
Live-Programmierung

http://www.openscad.org/

HTML – Hypertext Markup Language

- HTML ist eine „Sprache“ zum Strukturieren und Vernetzen von Dokumenten
- HTML kennt
  - ✔ Normalen Text
  - ✔ Strukturanweisungen (sog. „Tags“), die auch Verweise („Links“) zu anderen Dokumenten, lokal und im Internet enthalten
  - ✔ Schnittstellen zu Programmen („Plugins“) und Programmiersprachen („Skripts“)
  - ✔ Erweiterungen, die Layout und interaktives Verhalten des Dokumentes festlegen (CSS, „Cascading Style Sheets“)

Die Wikipedia-Seite zu HTML enthält auch eine Kurzeinführung zu den HTML-Tags.

Empfohlene Literatur

- Sprach-Referenz:
  - ☞ http://de.selfhtml.org/html/referenz/
- Umfangreicher Kurs zum Selbststudium:
  - ☞ http://de.selfhtml.org/

Grundstruktur HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
 <HEAD>
  <TITLE>Mein erstes HTML-Dokument</TITLE>
 </HEAD>
 <BODY>

Hier ist ein unsichtbarer Kommentar:

<!-- Dies ist ein Kommentar, er erscheint nicht in der Ausgabe,
     sondern nur im HTML-Quelltext! -->

Den Quelltext der Seite sieht man bei den meisten Browsern mit Tastenkombination
Steuerung-U.

Text wird der Darstellung automatisch umgebrochen und linksbündig formatiert.

Mehrfache          Leerzeichen          werden        als       eins    dargestellt.

Mehrfache Zeilenumbrüche

ebenfalls.
 </BODY>
</HTML>
```

HTML-Tags

<head>...</head>	Abschnitt mit „unsichtbaren“ Informationen über das Dokument
<title>...</title>	Text für die Titelleiste, nur innerhalb des Headers
<html>...</html>	Rahmt das gesamte Dokument ein
<body>...</body>	Block, der den „sichtbaren“ Teil des Dokumentes enthält
<h1 align=center>...</h1>	Überschrift Ebene 1, zentriert
	Verweis auf Dokument im Internet
<img src=“adresse-bild“>	Bild an dieser Stelle einblenden
...	

HTML-Tags

Listen

…	Unsortierte Liste
…	Nummerierte Liste
Text...	Listenelement (in beiden)
<dl> … </dl>	Beschreibungsliste
<dt>	Stichwort in der DL
<dd>	Beschreibung in der DL

HTML-Tabellen

```
<table border=1>
 <tr>
  <th>Überschrift Spalte 1</th><th>Überschrift Spalte 2</th>
 </tr>
 <tr>
   <td> Zeile 1, Spalte 1</td><td>Zeile 1, Spalte 2</td>
 </tr>
 <tr>
   <td> Zeile 2, Spalte 1</td><td>Zeile 2, Spalte 2</td>
 </tr>
 <tr>
   <td> Zeile 3, Spalte 1</td><td>Zeile 3, Spalte 2</td>
 </tr>
</table>
```

Überschrift Spalte 1	Überschrift Spalte 2
Zeile 1, Spalte 1	Zeile 1, Spalte 2
Zeile 2, Spalte 1	Zeile 2, Spalte 2
Zeile 3, Spalte 1	Zeile 3, Spalte 2

(Pre-)Formatierte Texte

- Trick A: Häufig werden Tabellen dafür verwendet, um Formularelemente oder allgemein, Teile des Dokumentes relativ zueinander zu positionieren.

- Trick B: Mit `<pre>` ... `</pre>` „eingerahmt“ erscheinen die Texte „beinahe“ immer genau dort, wo sie im HTML-Quelltext eingegeben wurden, sind aber gleichzeitig im „Schreibmaschinen-Zeichensatz“.

- Beides gilt als „schlechter Stil“, die moderne Variante zur absoluten oder relativen Positionierung von Teilen des Dokumented sind die ☞ „Cascading Stylesheets“ (CSS), die wir aber in der Vorlesung nicht im Detail behandeln werden.

Kodieren von Sonderzeichen – HTML Entities (1)

- Problem 1: Der Zeichensatz, in dem der Dokumentenquelltext geschrieben ist, entspricht nicht unbedingt dem Zeichensatz, den der Browser erwartet.
  → z.B. Fehler bei der Darstellung von Umlauten

- Problem 2: Für manche Sonderzeichen gibt es keine Taste auf der Tastatur, z.B.: π ğ ø Å

- Problem 3: Das Kleiner-Zeichen < und das Kaufmanns-UND & haben in HTML eine spezielle Bedeutung und können nicht „direkt“ ausgegeben werden.

Kodieren von Sonderzeichen – HTML Entities (2)

Lösung: Kodierung als „HTML-Entity“:

&Name; oder **&#Nummer;**

HTML-Quelltext	Wirkung	Merken mit
<	<	„Less Than“
>	>	„Greater Than“
&	&	„Ampers And“
ä Ä	ä Ä	„A Umlaut“
ö Ö	ö Ö	„O Umlaut“
ü Ü	ü Ü	„U Umlaut“
ß	ß	„S Z Ligatur“ (Scharfes S)
€	€	„Euro“

Türkische Sonderzeichen: Siehe hier.
Kyrillische Zeichen (Griechisch, Russisch): Siehe hier.

Zeichenkodierung des gesamten Dokuments

Im Header des HTML-Dokuments kann der Zeichensatz, der für die Darstellung verwendet werden soll (und in dem das Dokument geschrieben ist), angegeben werden (s.a. https://www.w3.org/International/questions/qa-html-encoding-declarations):

```
<HEAD>
 <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8">
</HEAD>
```

HTML-Formulare

- Formulare erlauben die Eingabe von Daten an festgelegten Stellen in einem HTML-Dokument.

```
<form action="programm-adresse">

            ...Formularelemente... *)

 <input type=reset value="Zurücksetzen">
 <input type=submit value="Abschicken">

</form>
```

*) Für die spätere *Auswertung* ist es außerdem wichtig, dass jedes Formular*element* eine eindeutige *Kennung* (Attribut **name=Kennung**) bekommt.

Formular-Elemente

Das einfache Texteingabe-Feld

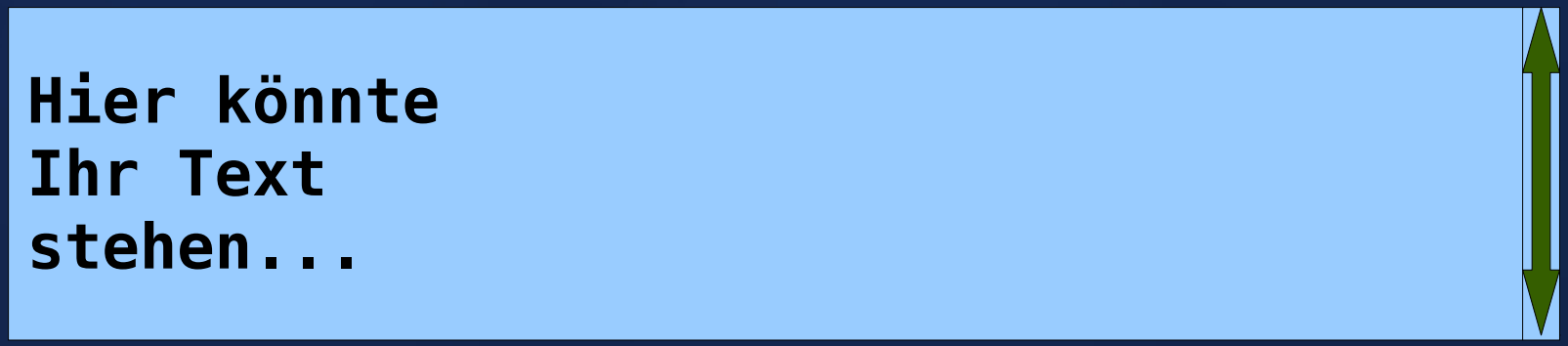
```
<input  name=Kennung
        type=text
        size=80
        maxlength=120
        value="Vorgabe">
```

Vorgabe

Formular-Elemente

Das mehrzeilige Texteingabe-Feld

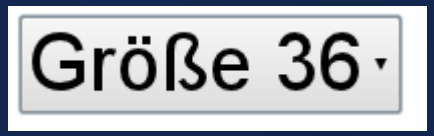
```
<textarea name=Kennung
          cols=40
          rows=20> Vorgabetext... </textarea>
```

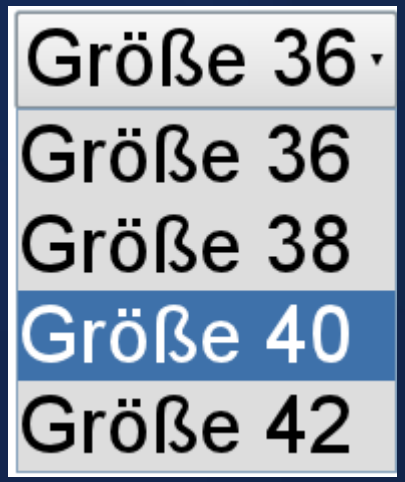


Formular-Elemente

- „Popup-Menüs“ sind eine platzsparende Art, aus einer Liste genau *ein* Element auszuwählen.

```
<select name=Kennung>
    <option value=36>Größe 36</option>
    <option value=38>Größe 38</option>
    <option value=40>Größe 40</option>
    <option value=42>Größe 44</option>
</select>
```





Formular-Elemente

- Die gleiche Konstruktion gibt es auch mit Mehrfach-Auswahl.

```
<select name=Kennung multiple>
 <option value=101>Papadams</option>
 <option value=102>Garlic Naan</option>
 <option value=103>Chicken Marsala</option>
 <option value=104>Firni</option>
</select>
```

Formular-Elemente

- Checkboxen sind „Schalter“, die an- und abgewählt werden können.

```
<input type=checkbox name=Kennung value=1>
   Bitte keine Werbung!<br>

<input type=checkbox name=Kennung value=2>
   Ich möchte am Gewinnspiel teilnehmen.<br>
```

☑ Bitte keine Werbung!
☐ Ich möchte am Gewinnspiel teilnehmen.

Formular-Elemente

- Radioboxen sind „Wechselschalter“, von denen immer nur *einer* ausgewählt sein kann.

```
<input type=radio name=Kennung value=a>
   Antwort A ist richtig!<br>
<input type=radio name=Kennung value=b>
   Antwort B ist richtig!<br>
<input type=radio name=Kennung value=c>
   Antwort C ist richtig!<br>
```

- Antwort A ist richtig!
- Antwort B ist richtig!
- Antwort C ist richtig!

ACTION!

- Noch können wir mit den HTML-Formularen wenig anfangen, denn hierzu müsste ein *Programm* die Auswertung der eingegebenen Daten übernehmen, und eine Ergebnisseite „zurückschicken“, oder zumindest irgend etwas *tun*.

- Das Attribut **`action=programm-adresse`** innerhalb des `<form>`-Tag teilt dem Browser mit, wohin er die Eingabedaten zu schicken hat, und woher er dann ggf. auch wieder ein Ergebnisdokument zurückgeschickt bekommt.

- Beim Themenkreis „internetorientierte Informatik“ werden wir auf die Programmiersprache ☞ PHP zurückgreifen, um Formulare auszuwerten und Ergebnisse darzustellen. Hierzu wird es notwendig sein, einen WWW-Server mit PHP-Unterstützung verfügbar zu haben!

☞ Javascript

- ...hat (so gut wie) nichts mit der Programmiersprache ☞ JAVA zu tun.
- ...ist in den meisten Browsern standardmäßig eingebaut, und unterstützt interaktive Funktionen, die vom Browser selbst durchgeführt werden.
- ...ist leider je nach Hersteller unterschiedlich implementiert (teilweise sehr gravierende Unterschiede in Syntax und Semantik von Dokumentstrukturen und Befehlen).
- ...wird oft eingesetzt, um „dynamische Webseiten“, „Rollover“-Funktionen und Browser-Bedienungserleichterungen, sogar einfache Spiele oder komplexe Grafikfunktionen, zu programmieren.*)

*) Wir beschränken uns in der Vorlesung auf die wichtigsten Funktionen für den „Hausgebrauch“.

Javascript skripten – das `<script>`-Tag

- Eine Möglichkeit, Javascript innerhalb einer HTML-Seite auszuführen, ist die `<script>...</script>`-Umgebung.

```
<script type="text/javascript">
  document.writeln("Hallo, Welt!");
</script>
```

Hierbei wird oft ein Trick angewendet, um mit dem „Kommentar-Tag“ für Browser, die kein Javascript können, das Programm „unsichtbar“ zu machen.

```
<script type="text/javascript">
   <!-- HTML-Kommentar
    // Javascript-Kommentar
   alert("Hallo, Welt!");
   // Ende. Ende HTML-Kommentar -->
</script>
```

Javascript als Attribut für HTML-Tags

- Einzelne Funktionen oder kürzere Javascript-Programme können auch direkt in andere Tags als Attribut hineingeschrieben werden. Die entsprechenden Attribute beginnen meist mit „**on**...“.

- In diesem Beispiel wird der „Wert des Textes“ in einem Abschnitt (Paragraph <p>) beim Klick darauf verändert. Man muss hier sehr darauf achten, mit den beiden „Zitatzeichen“ " und ' nicht durcheinander zu kommen. (" über der 2 und ' neben der Eingabetaste)

```
<p onClick="this.firstChild.nodeValue='Danke.'">
  Klickst Du hier!
</p>
```

Zugriff auf Formulardaten mit Javascript

```
<FORM name=Formular>
 <input type=text name=zahl>²
 <input type=button
        onClick="quadrat(); return false;"
        value=" = ">
 <input type=text name=ausgabe readonly>
</FORM>

<SCRIPT type="text/javascript">
 function quadrat(){
  var zahl = document.Formular.zahl.value;
  var ergebnis = zahl * zahl;
  document.Formular.ausgabe.value = ergebnis;
}
</SCRIPT>
```

if ... und else ... in Javascript

```
if(Bedingung) {
   Anweisungen, wenn Bedingung WAHR (true) ist
}
else {
   Anweisungen, wenn Bedingung FALSCH (false) ist
}
```

Beispiel:

```
if(document.Formular.password.value == "geheim"){
  document.writeln("Willkommen, Dr. Jones");
}
else {
  alert("Zugriff verweigert");
}
```

Javascript-Funktionen schreiben

- Eine Funktion ist eine Folge von Befehlen, die unter einem „Namen“ zusammengefasst wird. Sie kann, wie eine mathematische Funktion, ein Ergebnis produzieren, das einer Variablen zugewiesen werden kann.

```
function funktion_name(Übergabeparameter)
 {
   Befehle, durch ; getrennt.
   return Ergebnis;
 }
```

Aufruf der Funktion (Beispiel):

```
 var Resultat = funktion_name(10);
```

Internet-Dienste – Das Client/Server Prinzip

- Server: Bietet einen oder mehrere Dienste an, z.B.
  - ✔ Zugriff auf Dateien (http, ftp, bittorrent, smb, nfs, ...)
  - ✔ Interaktive Informationsdienste wie WWW (http, https)
  - ✔ Interaktive Rechnerzugänge, Teleworking (RDP, VNC, SSH)
  - ✔ Synchrone und asynchrone Nachrichtendienste (E-Mail/smtp, Chat/irc/icq/..., Telefonie/voip)
  - ✔ Video- und Audio-Live-Übertragung (WebRTC z.B. http://palava.tv oder rtsp:// (Video-Bradcast))

- Client: Nutzt die Dienste über ein Netzwerk, benötigt entsprechende Zugangssoftware für jeden Dienst:
  - ✔ WWW-Browser
  - ✔ E-Mail Programm (falls nicht über WWW)
  - ✔ Chat-Programme (pidgin, irc, ...)
  - ✔ Videokonferenzen (ekiga, skype)
  - ✔ Remote-Sessions (VNC/RDP-Client, SSH, Webbasierte Dialogdienste, Clients für Online-Spiele, WebRTC...)

„Connections“

- Der von allen Internet-Teilnehmern verwendete Standard heißt **TCP/IP** und wird von (fast) allen netzwerkfähigen Betriebssystemen verstanden.

- Eine Internet-Verbindung hat **5 Kenngrößen**:
  - ✔ Client IP-Adresse (z.B. `192.168.0.1`)
  - ✔ Server IP-Adresse (z.B. `85.214.68.145`)
  - ✔ Client Netzwerkport (beliebig, aber nur einer pro Verbindung)
  - ✔ Server Netzwerkport (i.d.R. durch den angesprochenen Dienst festgelegt, z.B., `80` für WWW, 25 für E-Mail usw.)
  - ✔ Netzwerkprotokoll
    - ✔ Untere Ebene: `TCP` (kontrolliert) oder `UDP` („verbindungslos“)
    - ✔ Anwendungsebene: Kommunikationsprotokoll, z.B. `http/https`, `ftp`, `ssh`, `rdp`, `smb`, `nfs`, `sip`, ...

Netzwerk-Infrastruktur

- Gültige und im gesamten Netzwerk eindeutige **IP-Adresse** (je eine für Client und Server).
- Verbindung über lokale Netze hinweg per **Gateway**s. Das dem eigenen Rechner nächste Gateway wird als „**Default Gateway**“ bezeichnet, und ist im LAN (Local Area Network) der sog. Router, im WLAN (Wireless LAN) der Accesspoint. Auch bei UMTS/GPRS steht ein Gateway beim Netzanbieter, eine abgeschirmte „Direktverbindung“ zwischen Client und Server im Internet existiert fast nie. → Mögliche Sicherheitsprobleme!
- Rechner verstehen nur IP-Adressen, Menschen können sich Rechner**namen** besser merken → **Nameserver** löst Namen in Adressen auf, ansonsten „Host nicht gefunden“-Fehler.

Netzwerk-Dienste installieren - WWW-Server

- Bei den meisten Linux-Distributionen ist der beliebteste Webserver – ☞ Apache – als Installationsoption mit dabei.
- Für Windows und einige andere Betriebssysteme gibt es ein Installationspaket ☞ XAMPP.
- Der Webserver sollte als Systemdienst gestartet werden. Linux: **`sudo /etc/init.d/apache2 start`**
- Nach erfolgreicher Installation läuft auf Port 80 der Webserver, und liefert Seiten über das http-Protokoll.
- URL: `http://localhost/` → Dateien unter `/var/www/html` (Linux), bzw. unter `C:\xampp\htdocs` (Windows)
- Erweiterungen:
  - ✔ Skriptsprache PHP (libapache2-mod-php5)
  - ✔ SQL Datenbank-Anbindung (php5-mysql)

Xampp Windows-Probleme lösen

- Port 80 im Firewall freigeben (evtl. auch Port 443)
- Falls ein anderes Programm bereits einen der beiden Ports belegt, Portnummern in der xampp-Konfiguration ändern (ggf. dann in allen Übungen `http://localhost:`**`neue_portnummer`**`/`... verwenden
- Im Dateimanager „Erweiterungen“ in der Ansicht erlauben, damit die Endung .txt, die manche Texteditoren automatisch hinzufügen, durch Umbenennen entfernt werden kann.

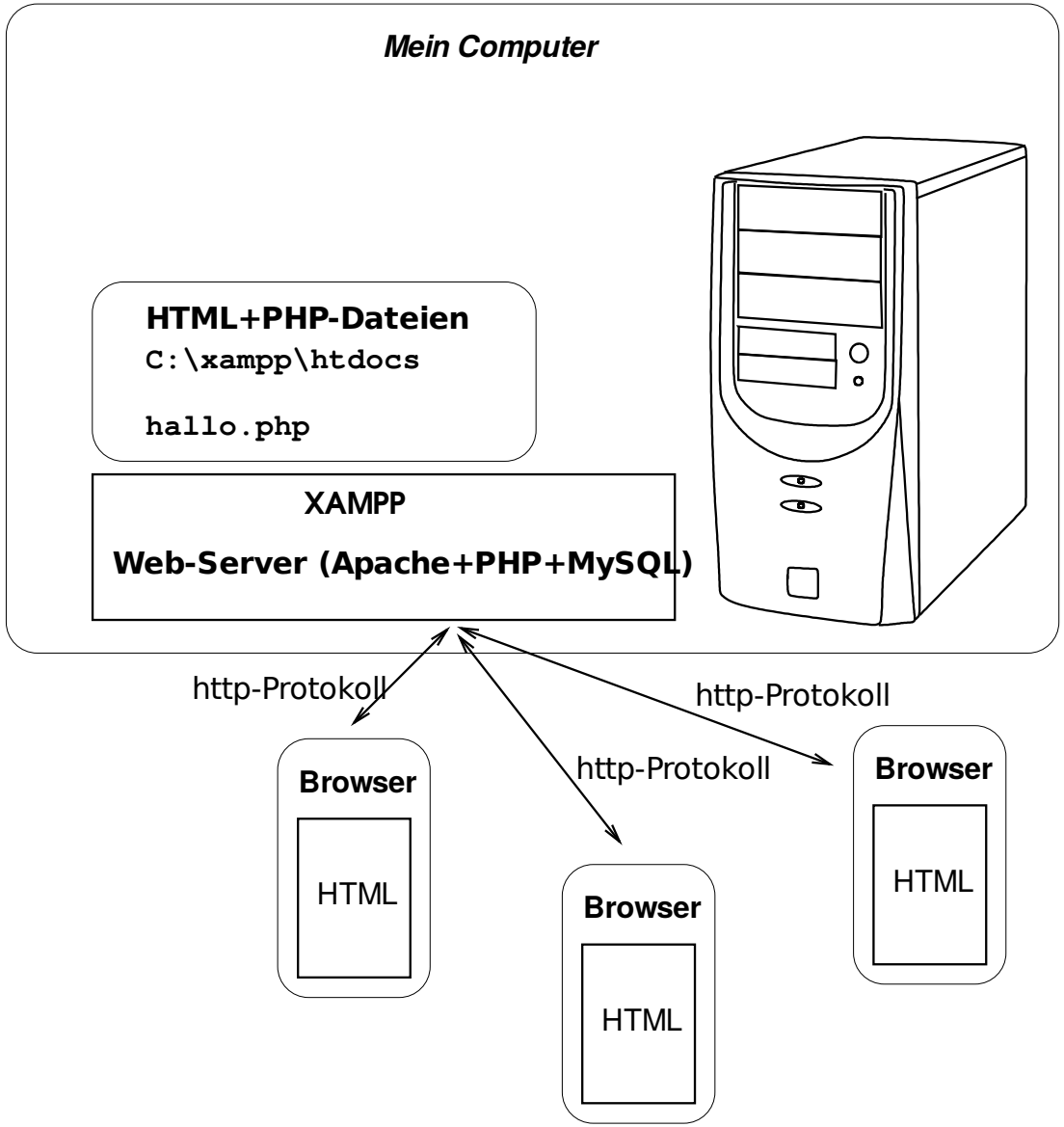
Übung

- Legen sie in dem Verzeichnis, das von Ihrem Webserver als „Document Root“ verwendet wird (Xampp unter Windows: **`C:\XAMPP\htdocs`**) ein Unterverzeichnis **`ainf`** an (Groß-/Kleinschreibung beachten).
- Kopieren Sie Ihre bisherigen HTML- und Javascript-Dateien in dieses Verzeichnis.
- Falls der Apache-Webserver noch nicht läuft, starten Sie ihn über das XAMPP-Kontroll-Panel
- Rufen Sie das Inhaltsverzeichnis übers Web ab: http://localhost/ainf/
  Beachten Sie: Der Pfad, unter dem die Dateien auf der Festplatte liegen, ist von dem Pfad, den Sie beim Web-Abruf im Browser eingeben, relativ unabhängig!

Apache-Webserver für PHP vorbereiten unter Linux

- Installation `libapache2-mod-php5`
- `sudo a2enmod php5`
- `sudo /etc/init.d/apache2 restart`
- `sudo chmod a+w /var/www/html`
- Anlegen einer Datei `index.php` unter `/var/www/html` und Abruf per Browser unter `http://localhost/index.php`

AINF 2018/19



http://adresse-mein-computer/hallo.php

Webhosting für Studenten

Tipp: Unter

http://stud.fh-kl.de/

können Sie Ihre eigenen Webseiten hochladen und betreiben. Ihr persönlicher Webspace ist anschließend unter

http://stud.fh-kl.de/~benutzername/

im Internet erreichbar.

PHP – Skripte in Webseiten

Übung

Legen Sie in Ihrem XAMPP-Dokumentenordner **C:\xampp\htdocs\ainf** eine neue Datei mit dem Namen **test.php** an, mit folgendem Inhalt:

```
<body>

  <H1 align=center>Ein einfaches PHP-Skript</H1>

  <?php echo "Hallo, Welt!"; ?>

  <?php phpinfo(); ?>

</body>
```

Rufen Sie nun die Seite im Browser unter http://localhost/ainf/test.php auf.

PHP-Elemente / Charakteristika

- Sieht (leider) sehr ähnlich aus wie JavaScript, aber durch Tags `<?php PHP-Skript... ?>` eindeutig gekennzeichnet, die den Browser nie erreichen sollten, da sie auf der Server-Seite bereits verarbeitet werden.
- Anweisungen **müssen** durch **;** abgeschlossen werden.
- Variablen beginnen mit dem **$**-Zeichen.
- Viele Funktionen vordefiniert und dokumentiert unter. www.php.net
- U.a. auch Datenbank-Anbindung (php-mysql) möglich.

Auswerten von Formularen

Bei **METHOD=POST** werden Formularfelder direkt an den Server verschickt, bei **METHOD=GET** über die Adresse (URL), und sind in der Browser-Adressleiste sichtbar. Dies hat auch Auswirkungen auf die Art der Abfrage in PHP.

Formular (Fragment):

```
<FORM METHOD="POST" ACTION="auswertung.php">
 <input name=antwort type=text>
 <input type=submit>
</FORM>
```

PHP-Skript „auswertung.php“ (Fragment):

```
<?php  echo "Sie haben ";
       echo $_POST[antwort];
       echo " eingegeben."; ?>
```

ÜBUNG

- Erzeugen Sie im Ordner, in dem Ihre PHP-Dateien liegen sollten (z.B. `C:\xampp\htdocs\ainf`) eine Datei **`formular_mit_auswertung.php`**

  In dieser soll nun sowohl das Eingabeformular (ein Feld reicht), als auch die Auswertung des Formulars (in PHP) stattfinden!

  Statt als ACTION=... wieder die gleiche Datei anzugeben, können Sie in diesem Fall den ACTION-Parameter einfach weglassen, dann „ruft die Datei sich selbst auf“, wenn auf „Abschicken“ geklickt wird.
- Fragment:
  bei FORM METHOD=GET (Standard)
  `echo $_GET['name_des_eingabefeldes'];`
  bei FORM METHOD=POST
  `echo $_POST['name_des_eingabefeldes'];`

Bedingungen

Auch in PHP können Entscheidungsabfragen und Alternativen stattfinden. Alles, was „ungleich null“, `true` oder „nicht leer“ ist, ist logisch WAHR.

```
if($_POST['antwort']) {
 echo "Antwort ist gesetzt."; }
else {
 echo "Antwort ist nicht gesetzt, oder 0"; }

if(stristr($_POST['antwort'], "suchtext") {
 echo "Antwort enthält suchtext.";
} else {
 echo "Antwort enthält suchtext nicht.";
}
```

Bessere Variante von „ist das Formularfeld x belegt?“

```
if( isset($_POST['formularfeldname']) ) …
```

→ Überprüft, ob das Feld überhaupt existiert, nicht nur ob es ungleich 0 ist oder Inhalt besitzt.

if … else … endif über größere Bereiche

Mit der Konstruktion

```
<?php if(Bedingung): ?>
```

HTML-Block 1

```
<?php else: ?>
```

HTML-Block 2

```
<?php endif; ?>
```

können Teile der HTML-Seite ein- und ausgeblendet werden, je nachdem, ob die Bedingung erfüllt ist oder nicht. Achtung: Schreibweise ist hier anders (ohne {…})!

Variablen in PHP

Variablen in PHP sind nicht streng typisiert, können also wie bei Javascript auch wahllos Zahlen oder Texte enthalten.

```
<?php
 $var1 = "1"; // Ein Text
 $var2 = 2;   // Eine Zahl
 echo "$var1 + $var2 = ";
 echo $var1 + $var2;
 echo "<br>";
 echo $var3; // Was wird hier wohl ausgegeben?
?>
```

Arrays (Datenfelder) in PHP

Bereits bekannt: `$_POST['formularfeldname']`
1. Es sind neben Zahlen in [..] auch Zeichenketten, z.B. `'eingabe'` oder `"eingabe"` ... erlaubt.
2. Es lässt sich zwar die ANZAHL der Elemente auslesen, aber grundsätzlich können beliebige, auch nicht belegte „Stellen“ im Array über die Indizes angesprochen werden.
3. Eine spezielle Art der for()-Schleife erlaubt das Auslesen von Inhalten eines Arrays nach Index:

```
foreach($_POST as $key => $value) {
 echo "Der Wert des Arrays \$_POST an der Stelle
$key ist $value.<br>\n";
}
```

→ Bildet `$_POST['key']` auf den enthaltenen `value` ab.

Daten speichern

- Um Formulardaten auf dem Server zu speichern, können einfache, für den Webserver schreibbare Dateien verwendet werden.

```
<?php
 $datei = fopen("datei.txt", "w");
 if($datei) {
  fwrite($datei, $text);
  fclose($datei);
 } else {
  echo "Kann Datei leider nicht öffnen!");
 }
?>
```

Das Speichern in Dateien hat gewisse Nachteile...

Dateirechte unter Linux und Windows

- Aus Sicherheitsgründen hat der Apache-Webserver normalerweise KEINE Schreibrechte auf Dateien in den fürs WWW freigegebenen Verzeichnissen.
- Unter Linux können die Dateirechte entsprechend geändert werden:
  `chmod a+w gaestebuch.txt`
  unter Windows auch, aber es ist evtl. komplizierter...
  Lösungsansatz: Schreibbare Dateien generell in ein Verzeichnis legen, das für alle schreibbar ist (z.B. `'C:\Windows\Temp'`), aber: Gefahr versehentlichen Löschens.
- Bei gleichzeitigem Schreibzugriff mehrere Programme auf eine Datei kann Chaos entstehen.
- Professioneller Lösungsansatz: Datenorganisation mit SQL-Datenbank

Datenbanken

- Eine ☞ Datenbank dient dem dauerhaften Speichern von Daten.
- Datenbanken stellen Mechanismen zum Anlegen, Löschen, Verändern und v.a. Zum effizienten Durchsuchen von Datensätzen zur Verfügung.
- Datenbanken können je nach Typ lokal (Festplatte) oder über ein Netzwerk angesprochen werden.
- Um auf Daten zuzugreifen, wird ein genau definierter Befehlssatz verwendet, der in verschiedenen Programmiersprachen unterstützt wird, beispielsweise ☞ SQL.
- Datenbanken, die Daten zueinander in *Beziehung* setzen (z.B. in Form von Tabellen), heißen ☞ Relationale Datenbanken.

SQL

- ☞ „Structured Query Language“, eine standardisierte Datenbanksprache zur Definition, Abfrage und Manipulation von Daten.
- Organisation in „Datenbanken“ und darin enthaltenen „Tabellen“.
- Eine „Tabelle“ stellt eine zweidimensionale Relation (=Beziehung) durch Namen und Werte dar. Verweise zwischen verschiedenen Tabellen sind durch „Schlüssel“ möglich.
- SQL erlaubt es, von einem Client-Programm aus mit Hilfe von SQL-Befehlen direkt mit der Datenbank zu „sprechen“.
- Verschiedene Programmiersprachen unterstützen SQL mit Hilfe von Funktionen, die den Sprachumfang erweitern (z.B. php5-mysql).

MySQL

- MySQL ist eine SQL-Datenbank auf Open Source Basis, und wie der Apache-Webserver für verschiedene Betriebssysteme verfügbar.
- Start unter Linux:
  `sudo /etc/init.d/mysql start`
- Kommandozeilen-Client (das „-p“ entfällt, wenn kein Passwort für den Datenbank-Administrator gesetzt ist):

Linux	Windows
`mysql --user=root -p`	`C:\xampp\mysql\bin\mysql --user=root`

-
```
mysql> \u mysql
Database changed

mysql> show tables;
+-------------------------+
| Tables_in_mysql         |
```

CREATE – Datenbanken erzeugen und benutzen

Syntax: CREATE DATABASE *name*;

Beispiel: Datenbank für den AINF-Kurs erzeugen

```
mysql> create database ainf;
Query OK, 1 row affected (0,00 sec)

mysql> use database ainf;
Database changed.
```

CREATE – Tabellen erzeugen

Syntax: CREATE TABLE name (feldname feldtyp, ...)

Beispiel: Tabelle für Benutzer-Kennungen

```
mysql> create table student
        (name varchar(80), matnr integer,
         semester integer);
```

- integer: Ganze Zahl
- varchar(anzahl): Zeichenkette mit konstanter Länge.

INSERT – Zeilen zu einer Tabelle hinzufügen

```
Syntax: INSERT INTO tabellen_name
                    (feldname1, feldname2, …)
             VALUES ("Wert 1", "Wert 2", …);

mysql> insert into student
               (name, matnr, semester)
        VALUES ("Klaus Knopper", 228813,
                 20);

Query OK, 1 row affected (0.05 sec)
```

SELECT – Daten auswählen und lesen

Syntax: SELECT feldname FROM tabelle_name
WHERE Kriterium;

Beispiel: Alle Zeilen der Tabelle „student“ anzeigen

```
mysql> select * from student;
+----------------+---------+----------+
| name           | matnr   | semester |
+----------------+---------+----------+
| Klaus Knopper | 228813  | 20       |
+----------------+---------+----------+
1 row in set (0.00 sec)
```

SELECT – Auswahl bestimmter Daten nach Kriterien

Beispiel: Suche nach Stichworten (Substrings) in einer Tabelle.

```
mysql> select name,semester
               from student
               where name like '%klaus%';

+----------------+----------+
| name           | semester |
+----------------+----------+
| Klaus Knopper | 22       |
+----------------+----------+
1 row in set (0.00 sec)
```

SELECT – Auswahl bestimmter Daten nach Kriterien

Beispiel: Suche nach Zahlenwerten in einer Tabelle.

```
mysql> select name,semester
             from student
             where semester > 7;

+----------------+----------+
| name           | semester |
+----------------+----------+
| Klaus Knopper | 22       |
+----------------+----------+
1 row in set (0.00 sec)
```

Auch möglich: = (gleich) > (größer) < (kleiner)
!= (ungleich)

UPDATE – Vorhandene Zeilen modifizieren

Syntax:
```
UPDATE tabelle_name SET feldname = 'Wert'
                        WHERE Kriterium;
```

Beispiel:

```
mysql> update student set name='Knopper, Klaus'
                    where name='Klaus Knopper';

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DELETE – Löschen von Tabellenzeilen

Syntax: `DELETE FROM tabelle_name WHERE Kriterium;`

Beispiel: Löschen der Zeile für den User „knopper“

```
mysql> delete from student where matnr = 228813;
Query OK, 1 row affected (0.00 sec)
```

Das WHERE-Kriterium ist mit dem des SELECT-Befehls identisch.

DROP – Tabellen vernichten

Syntax: DROP TABLE tabelle_name;

WARNUNG: Es gibt keine Rückfrage, ob eine umfangreiche Tabelle wirklich gelöscht werden soll!

```
mysql> drop table student;
Query OK, 0 rows affected (0.00 sec)
```

PHPMyAdmin

- Um Tabellen mit einem graphischen Frontend und ohne Detailkenntnis von SQL bearbeiten zu können, wird oft das PHP-basierte **PHPMyAdmin** auf Apache2 mit PHP-MySQL eingesetzt.
- Nach der Installation des phpmyadmin-Paketes unter Linux (`sudo apt-get install phpmyadmin`) bzw. in xampp unter Windows steht unter der Adresse
  `http://localhost/phpmyadmin`
  eine graphische MySQL-Administrationsumgebung zur Verfügung.
- Um auch ohne Passwort in `phpmyadmin` arbeiten zu können (was natürlich unsicher ist), kann unter Linux in der Datei `/etc/phpmyadmin/config.inc.php` die Zeile
  `$cfg['Servers'][$i]['AllowNoPassword'] = TRUE;`
  aktiviert werden.

Datenbank-Zugriff mit PHP-Mysql *(bis PHP 5)*

- Erweiterung, die zu PHP Funktionen zum Zugriff auf MySQL-Datenbankserver hinzufügt.
- SQL-Kommandos werden über sog. **queries** verschickt, und Abfragen der Ergebnisse speichern dieselben in Arrays.

```
<?php
 mysql_connect("localhost","root","passwort");
 @mysql_select_db("mysql")
  or die("Unable to select database");
 mysql_query("CREATE TABLE besucher (name
varchar(80), email varchar(80));");
 mysql_close();
?>
```

Datenbank-Zugriff mit PHP-Mysqli *(ab PHP 5)*

- Erweiterung, die zu PHP Funktionen zum Zugriff auf MySQL-Datenbankserver hinzufügt, **jetzt mit Unterstützung für mehrere parallele Server**.
- SQL-Kommandos werden, Server-bezogen, über sog. **queries** verschickt, und Abfragen der Ergebnisse speichern dieselben in Arrays.
- Die alten „`mysql_`...“ (s. vorige Folie) Befehle ***werden ab PHP7 nicht mehr unterstützt***!
- mysql/mysql**i**-Syntax ist nur teilweise kompatibel.
- Klausurrelevant ist die **neue Syntax** (**`mysqli_`**...).

```
<?php
 $db = mysqli_connect("localhost",
  "username","userpasswort", "userdatenbank");
 mysqli_query($db,"CREATE TABLE besucher (name
varchar(80), email varchar(80));");
 mysqli_close($db);
?>
```

Verbindung zur Datenbank herstellen

mysqli_connect(…) meldet das Programm beim MySQL-Server mit den angegebenen Benutzerdaten an (Achtung: Benutzer vorher einmalig einrichten!) und wechselt gleich in die angegebene Datenbank. Fehler können mit weiteren Funktionen erkannt werden.

```
<?php
 $db = mysqli_connect("localhost",
  "username","userpasswort", "userdatenbank");

   if (mysqli_connect_errno()) // Fehlernummer?
   {
    echo "Verbindung zu MySQL gescheitert: ";
    echo mysqli_connect_error(); // Fehlermeldung
    die(); // Programm beenden
   }
?>
```

mysqli_query()

`mysqli_query($server, "`*`SQL-Kommando`*`")` schickt ein SQL-Kommando an den MySQL-Server und liefert einen Ergebnis-*Vektor* zurück, der mit anderen Funktionen abgefragt werden kann.

```
$result = mysqli_query($db, "SELECT * from
Tabelle;");
```

`$result` enthält selbst keine Ergebnisse, ist aber eine *Referenz* auf das Ergebnis, und ermöglicht das „Abholen“ der tatsächlichen Ergebniszeilen der SQL-Abfrage, hierzu gibt es weitere Kommandos (nächste Folie).

mysqli_query()-Resultat verarbeiten

```
echo "Das Ergebnis hat ";
echo mysqli_num_rows($result);
echo " Zeilen.";

while( $content = mysqli_fetch_assoc($result) ){ // Schleife

    // Bestimmte Felder ausgeben
    echo $content['field1_name']; echo "<br>“;
    echo $content['field2_name']; echo "<br>“;

    // Oder: alle Felder ausgeben
   foreach($content as $key => $value)
      echo "Feld $key enthält $value. <br>";

 } // Ende Schleife
```

Daten in DB-Tabellen einfügen mit PHP Mysqli

```
<FORM METHOD=POST>
 Name: <input name=name type=text><br>
 E-Mail: <input name=email type=text><br>
 <input type=submit value="Abschicken">
</FORM>
...

<?php
 $name=$_POST['name']; $email=$_POST['email'];
 $db = mysqli_connect("localhost",
  "username","userpasswort", "userdatenbank");
 $query = "INSERT INTO besucher (name, email) VALUES
('$name','$email');";
 mysqli_query($db, $query);
 mysqli_close($db);
?>
```

MySQL-Fehler in PHP melden

```
$db = mysqli_connect("localhost",
"username","userpasswort",
"userdatenbank");

$sql = "SELECT * FROM tabelle;" ;

$result = mysqli_query($db, $sql);

if( ! $result ) { // Gab es kein Ergebnis?
  echo "<font color=red>Datenbank-Fehler: ";
  echo mysqli_error($db); // Fehler zeigen
  echo "</font>";
}
```

Was läuft wo?

	Client	Server
HTML	Darstellung im Browser, lokale HTML-Dateien oder entfernte Dateien..	Liefert auf Abruf Dateien via http-Protokoll.
Javascript	Im Browser integrierte „aktive“ Skriptsprache, oft als Abschnitt in HTML-Dateien.	
PHP		Wird vom http-Server ausgeführt (z.B. Apache mit PHP-Modul). Nur das Ergebnis der Ausführung wird an den Client übertragen.
MySQL	SQL-Client, z.B. MySQL-Kommandozeilentool oder phpmaydmin/PHP-Skripte	MySQL-Server, lokal oder auf entferntem Rechner.

Entity Relationship Modelle

- ☞ ERM = Graphische Darstellung von Objekten, ihren Eigenschaften und Beziehungen zueinander.
- „Wirtschaftsinformatische Alternative“ zu ☞ UML.

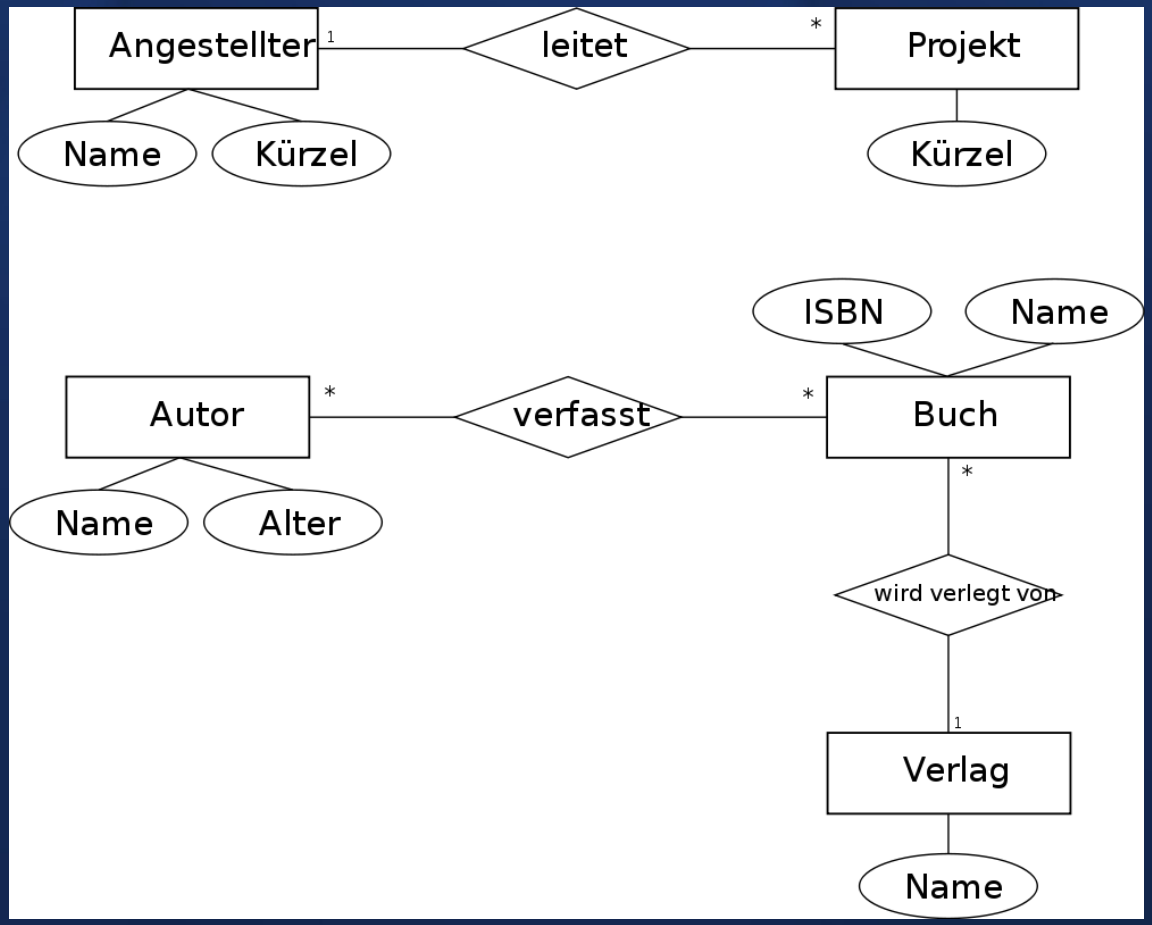
- Entität (Entity): Individuum, ein reales Objekt, z.B. eine Person, ein Projekt, ein Gegenstand.
- Beziehung (Relationship): Verknüpfung oder Zusammenhang zwischen Entitäten, beispielsweise Anbieter → Kunde, Projektmanager → Projekt, Kunde → Konto oder Bestellung.
- Eigenschaft (Attribute): Basisinformationen zum Objekt, z.B. Bezeichnung, Größe, Farbe, Alter.
- Entitäten und Beziehungen werden oft in Kategorien/Typen eingeteilt, z.B. „***Angestellter***“, bzw. Person A ***leitet*** Projekt X.

Spezielle Begriffe aus ERM

- ***Kardinalität***: Anzahl von möglichen Instanzen bzw. Beziehungen (Minimum, Maximum), die eine Entität besitzen kann.
- ***Reflexivität***: Beziehung zwischen Entitäten des gleichen Typs, z.B. „Teil A wird verwendet in Teil B“.
- ***Komplexität*** (oder Grad): Gibt an, aus wie vielen Entitätstypen eine Beziehung zusammengesetzt ist. Meistens 2, aber in manchen Fällen auch 3 und mehr.

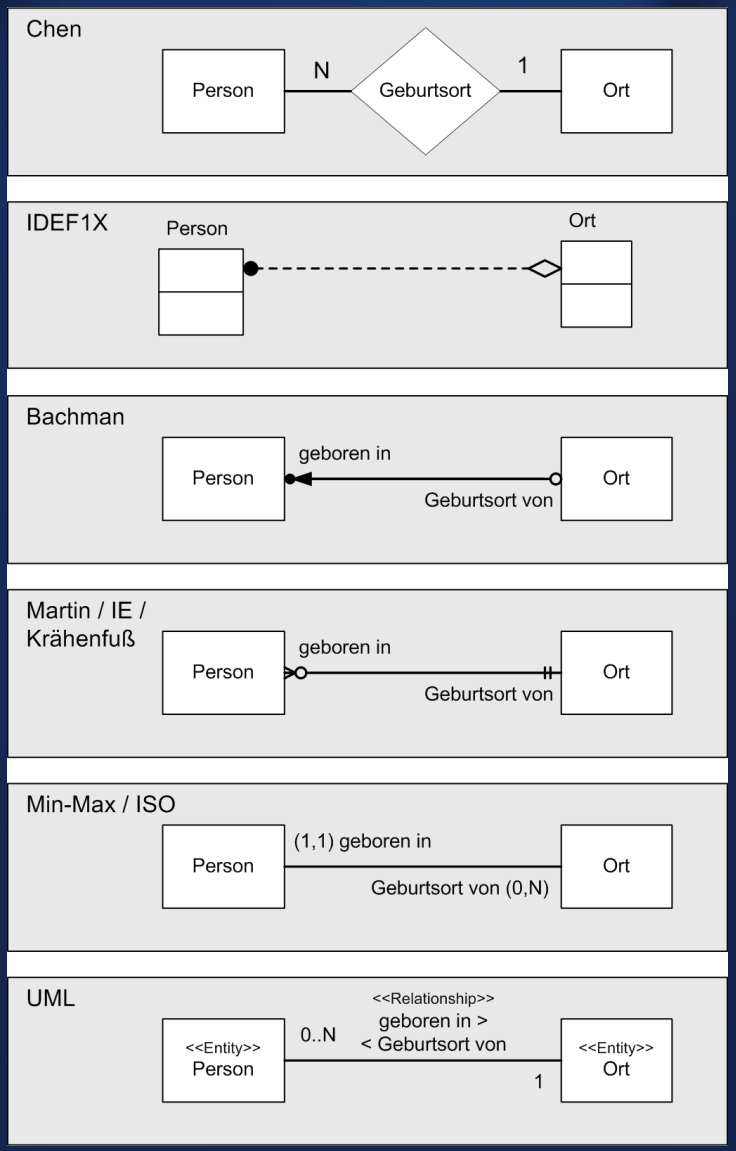
ERM-Diagramme

- Beispiel:



- Verschiedene Varianten sind gebräuchlich, s.a.
  - ☞ ERM-Diagramme auf Wikipedia

ERM-Diagramme – verschiedene Darstellungen



Wir verwenden die Darstellung nach Chen.

ERM-Diagramme

- (Beispiel-Diagramme von Wikipedia übernommen): Trotz der unterschiedlichen Darstellungsweisen sollen ERM-Diagramme die Zusammenhänge immer leicht verständlich erfassen.
- Wichtiger als eine 100% „standardisierte“ Darstellung ist die Verständlichkeit, und die Vermeidung von Missverständnissen → Klare und eindeutige Beschriftung verwenden!
- ERM- oder UML-Diagramme können als Grundlage verwendet werden, um Datenstrukturen in verschiedenen Programmiersprachen, oder um Tabellen in Relationalen Datenbanken zu entwerfen. Anderes Beispiel: Erstellen von Java-Klassen aus UML-Skizzen mit ☞ argouml.

Zeichenprogramme für ERM und UML

- argouml (Open Source)
- MS Visio (Proprietär, kommerziell)
- ARIS (proprietär, Freeware)
- Dia (Open Source)
- Kivio (Open Source)
- Inkscape (Open Source)
- OpenOffice/LibreOffice Draw (Open Source)

Entwicklung einer Datenbank-Struktur aus einem ERM

Beispiel: (1) KUNDE → BESITZT → (1..*) KONTO

d.h. „Ein Kunde besitzt mindestens ein Konto“.

KUNDE hat die Attribute NAME, ADRESSE, KUNDE_ID, KONTO die Attribute KONTO_ID,BANK, BIC, IBAN (Tafelskizze!).

→ Wie stellt man den Zusammenhang als Tabelle dar?

Lösung: Verwenden einer *Koppeltabelle* für die Relation „BESITZT“:

KUNDE_ID	KONTO_ID
10	2001
10	1002

Weitere Themen

- Sicherheit: WPA2-Krack-Attack, Meltdown und Spectre
- Bitcoin und Blockchain
- 3D Konstruktion mit OpenSCAD
- Sicherheit: CPU-Lücken Meltdown und Spectre

→ Separate Foliensätze/Handouts beachten (klausurrelevant)