



INTERACTIVE
MEDIA LAB
DRESDEN

Übung zur Vorlesung im Wintersemester 2017/2018

2 Übung Web- und Multimedia-Engineering

Aufgabenstellung A2 XML und PHP, thematische Einführung

- *Überblick (Terminplan)*
- 1. Aufgabenstellung A2: XML und PHP
- 2. Anwendung von serverseitigen Technologien:
 - a) XML und Freunde
 - b) PHP (**P**HP: **H**ypertext **P**reprocessor)
- 3. Hilfreiches, Tipps und Links

Woche	Datum	Übungsthema /-inhalt	Folien	Materialien
KW 41	9./10.10.	keine Übung (erste Vorlesungswoche)		
KW 42	16./17.10.	Einführung Aufgabenstellung A1: HTML + CSS + JS 		<ul style="list-style-type: none"> ■ Screenshots ■ Logo ■ Daten (CSV)
KW 43	23./24.10.	Konsultation		
KW 44	30.10.	Mo. Konsultation, Di. 31.10. keine Übung (Reformationstag)		
KW 45	6./7.11.	Abgabe A1		
KW 45	6./7.11.	Einführung Aufgabenstellung A2: PHP + XML		Materialien für A2
KW 46	13./14.11.	Lösung A1 und Konsultation		
KW 47	20./21.11.	Abgabe A2		
KW 47	20./21.11.	Einführung Aufgabenstellung A3: Webservices Node.js + AJAX		Materialien für A3
KW 48	27./28.11.	Lösung A2 und Konsultation		
KW 49	4./5.12.	Konsultation		
KW 50	11./12.12.	Abgabe A3		
KW 50	11./12.12.	Einführung Aufgabenstellung A4: Anwendungsbeispiel		Materialien für A4
KW 51	18./19.12.	Lösung A3 und Konsultation		
KW 52	25./26.12.	Jahreswechsel (keine Übung)		
KW 1	1./2.1.	Jahreswechsel (keine Übung)		
KW 2	8./9.1.	Konsultation		
KW 3	15./16.01.	Konsultation		
KW 4	22./23.01.	Abgabe A4		
KW 4	22./23.01.	Abschluss, Feedback und Fragen		
KW 5	29./30.01.	Lösung A4 und offene Fragen (nur nach Anmeldung)		
Woche	Datum	Übungsthema /-inhalt	Folien	Materialien

https://imld.de/study/teaching/ws_17-18/wme_17-18/ueb-wme_17-18/


Terminplan / Ablauf

- **A1:** Grundlagen client-seitige Technologien: HTML5, CSS3 & JS
 - Grundgerüst für eine Website als Interface für world_data
- **A2:** Grundlagen server-seitige Technologien: XML und PHP
 - CSV sowie XML Transformation, Grundlagen PHP Serverkomponente
- **A3:** Erweiterung server-seitige Technologien: Node.js & AJAX
 - Erstellung eines REST-Services, Abfragen durch den Client via AJAX
- **A4:** Anwendungsfall – Visualisierung mit D3.js & Leaflet
 - Visualisierung von Daten mittels interaktiver Bar Charts und Karten



Teil 1

Aufgabenstellung A2: XML und PHP

- Basis ist
 - die in A1 erstellte Webseite oder Leer-Vorlage aus den Materialien
 - Aufgabe umfasst insgesamt 3 Arbeitspakete
 1. Serverseitiges Parsen einer CSV Datei
 2. Eingelesene Datenstruktur als XML Datei speichern
 3. Verwenden der erstellten XML Datei und Transformation via XSLT in valides HTML5
-  *Alle drei Arbeitspakete sollen nacheinander einzeln ausführbar sein (Navigationselemente A2-Parse, A2-Save, A2-Print).*



≡ A1 - Table

≡ A2 - Parse

≡ A2 - Save

≡ A2 - Print

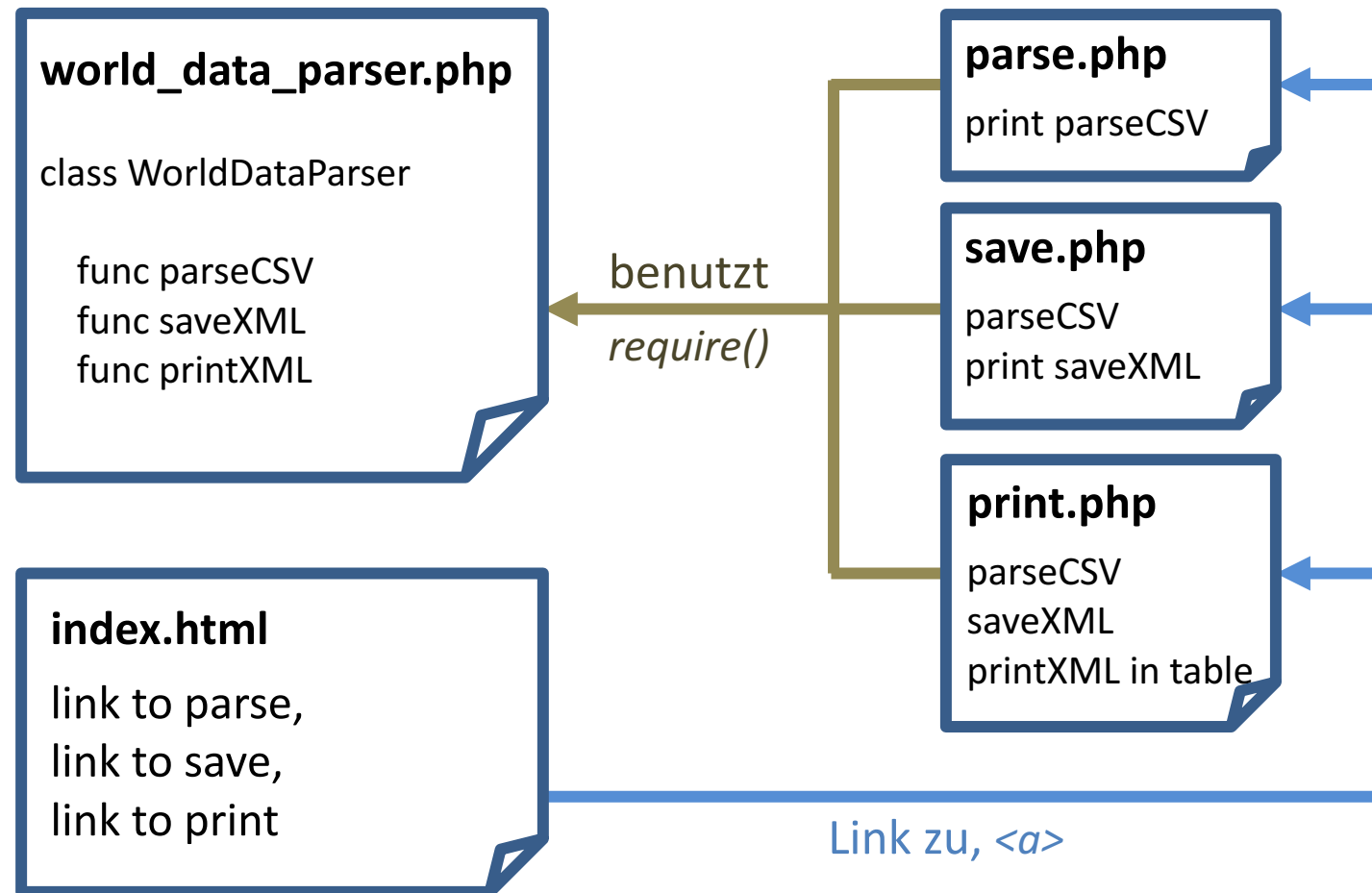
≡ A3 - REST

≡ A4 - Vis

World Data Overview ...

ID	Country	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
----	---------	-------------	-------------	-------------	-------------	-------------

- Schematischer Zusammenhang der PHP-Funktionen



1. Serverseitiges Parsen einer CSV Datei

- ☐ Klasse `WorldDataParser` in der PHP-Datei `world_data_parser.php` anlegen
- ☐ Funktion `parseCSV()` in Klasse `WorldDataParser` definieren
- ☐ Der Funktion wird ein Pfad übergeben (Parameter)
- ☐ Einlesen der CSV-Datei mit z.B. `fgetcsv()`

```
array fgetcsv(resource $handle, integer $length, string $delimiter)
```

<http://php.net/manual/de/function.fgetcsv.php>

- ☐ Funktion soll ein Array mit den Daten zurückliefern (Return)
- ☐ Die Datei `parse.php` soll die Funktion `parseCSV()` aufrufen und die zurückgegebene Datenstruktur in einem HTML `<pre>` Element anzeigen

2. Serverseitiges erstellen einer XML Datei

- ☐ Funktion `saveXML()` in Klasse `WorldDataParser` definieren
- ☐ Der Funktion wird ein Array mit den Daten übergeben (Parameter)
- ☐ Daten in XML umwandeln
- ☐ XML Datei soll im gleichen Ordner als `world_data.xml` gespeichert werden
- ☐ Die Funktion soll einen `boolean` Wert zurück liefern (Result), je nachdem ob das Schreiben erfolgreich war (true) oder nicht (false)
- ☐ Die Datei `save.php` soll die Funktionen `parseCSV()` und `saveXML()` aufrufen und den Rückgabewert von `saveXML()` durch eine von Menschen lesbare, verständliche Statusmeldung ausgeben

```
<?xml version="1.0" encoding="UTF-8"?>
<Countries>
  <Country>
    <id>001</id>
    <name>Brazil</name>
    <birth>16.405</birth>
    <cell>90.01936334</cell>
    <children>1.862</children>
    <electricity>2201.808724</electricity>
    <gdp_per_capita>
      4424.758692
    </gdp_per_capita>
    <gdp_per_capita_growth>
      -1.520402823
    </gdp_per_capita_growth>
    <inflation>8.228535058</inflation>
    <internet>39.22</internet>
    <life>74</life>
    <military>1.615173655</military>
    <gps_lat>-14.235004000</gps_lat>
    <gps_long>-51.925280000</gps_long>
  </Country>
  ...
</Countries>
```

3. Transformation von XML zu HTML (XSLT-Verarbeitung)

- ☐ Funktion `printXML()` in Klasse `WorldDataParser` definieren
- ☐ Der Funktion wird der Pfad zu einer XML-Datei sowie der Pfad zu einem XSLT Stylesheet übergeben (Parameter)
- ☐ Die Funktion soll das XML Dokument via XSLT in eine valide HTML-Tabelle überführen und zurückgeben (Result)
- ☐ Dazu soll der PHP-XSLT-Prozessor verwendet werden
<http://php.net/manual/en/class.xsltprocessor.php>
- ☐ Die Datei `print.php` soll:
 - ☐ `parseCSV()`, `saveXML()` und `printXML()` aufrufen
 - ☐ die zurückgegebenen Daten von `printXML()` sollen in einer Tabelle wie in Aufgabe 1 (A1) ausgegeben werden

■ Allgemeine Kriterien

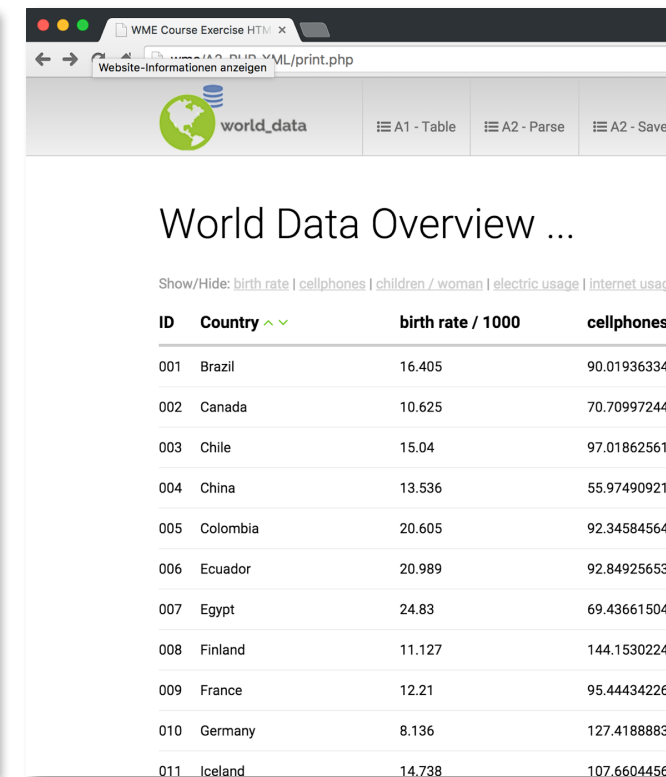
- ☐ Dateikodierung (Encoding): UTF-8 und *Unix-LF (Zeilenende)*
- ☐ Dokumentation von PHP- und XML-Lösung im Code
- ☐ Relative Adressierung verwenden (!), da Lösung in Unterordner ausgeführt wird, bspw. */root/Team_??/*

```
wme/A2_PHP-XML/parse.php

Array
(
    [0] => Array
        (
            [id ] => 001
            [name ] => Brazil
            [birth rate per 1000] => 16.405
            [cell phones per 100] => 90.01936334
            [children per woman] => 1.862
            [electricity consumption per capita] => 2201.808724
            [gdp_per_capita] => 4424.758692
            [gdp_per_capita_growth] => -1.520402823
            [inflation annual] => 8.228535058
            [internet user per 100] => 39.22
            [life expectancy] => 74
            [military expenditure percent of gdp] => 1.615173655
            [gps_lat ] => -14.235004000
            [gps_long] => -51.925280000
        )
    [1] => Array
        (
            [id ] => 002
            [name ] => Canada
            [birth rate per 1000] => 10.625
            [cell phones per 100] => 70.70997244
            [children per woman] => 1.668
            [electricity consumption per capita] => 15119.76414
            [gdp_per_capita] => 25069.86915
            [gdp_per_capita_growth] => -3.953353186
            [inflation annual] => 2.944408564
            [internet user per 100] => 80.17086651
            [life expectancy] => 80.9
            [military expenditure percent of gdp] => 1.415710422
            [gps_lat ] => 56.130366000
            [gps_long] => -106.346771000
        )
    [2] => Array
        (
            [id ] => 003
            [name ] => Chile
            [birth rate per 1000] => 15.04
            [cell phones per 100] => 97.01862561
            [children per woman] => 1.504
            [electricity consumption per capita] => 127.4188883
            [gdp_per_capita] => 127.4188883
            [gdp_per_capita_growth] => 12.21
            [inflation annual] => 8.136
            [internet user per 100] => 14.738
            [life expectancy] => 107.6604456
            [military expenditure percent of gdp] => 107.6604456
            [gps_lat ] => 107.6604456
            [gps_long] => 107.6604456
        )
)
```

```
wme/A2_PHP-XML/save.php

XML Savestatus: erfolgreich (1)
```



WME Course Exercise HTML x

Website-Informationen anzeigen

world_data

A1 - Table A2 - Parse A2 - Save

World Data Overview ...

Show/Hide: birth rate | cellphones | children / woman | electric usage | internet usage

ID	Country ^ v	birth rate / 1000	cellphones
001	Brazil	16.405	90.01936334
002	Canada	10.625	70.70997244
003	Chile	15.04	97.01862561
004	China	13.536	55.97490921
005	Colombia	20.605	92.34584564
006	Ecuador	20.989	92.84925653
007	Egypt	24.83	69.43661504
008	Finland	11.127	144.1530224
009	France	12.21	95.44434226
010	Germany	8.136	127.4188883
011	Iceland	14.738	107.6604456

- Erlaubte Hilfsmittel: sämtliche PHP-Funktionalität nutzbar
 - Keine weiteren PHP-Frameworks erlaubt!
- Testen: XAMPP (bspw. Version 7.1.10 mit **PHP >= 7.1.10**), Firefox (aktuelle Version)
- Abgabe: Montag/Dienstag, **20.11.2017 & 21.11.2017** 

Teil 2

Anwendung von serverseitigen Technologien: XML und Freunde, PHP

- XML-Dokument
 - Logischer Aufbau: Prolog + Wurzelement
 - Prolog
 - Streng genommen optional, typischerweise mindestens eine XML-Deklaration
 - Wurzelement
 - i.W. Elemente, Attribute und textuelle Inhalte
 - Enthält gesamte Daten des Dokuments
 - Max. ein Wurzelement, alle weiteren Inhalte darin

```
<?xml version="1.0"?>
<quiz>
  <frage>
    Wer war der fünfte
    deutsche Bundespräsident?
  </frage>
  <antwort>
    Karl Carstens
  </antwort>
  <!-- Anmerkung: Wir
    brauchen mehr Fragen -->
</quiz>
```

Quelle: [\[http://de.wikipedia.org/wiki/Xml\]](http://de.wikipedia.org/wiki/Xml)



■ Wohlgeformtheit

- Das Dokument besteht aus mindestens einem Element
- Es gibt genau ein Wurzelement
- Alle weiteren Elemente haben ein übergeordnetes Element, indem sie beginnen und auch enden
- Alle geöffneten Elemente werden geschlossen
- Alle Entities sind deklariert (bis auf amp, lt, gt, apos und quot)

■ Gültigkeit

- Ein XML-Dokument ist gültig, wenn es eine zugehörige Grammatik gibt und es dieser entspricht
- Jedes gültige XML-Dokument ist automatisch wohlgeformt

- XPath: XML Path Language, zur XML-Pfadbeschreibung
 - Dient der Adressierung beliebiger Knoten (*nodes*) oder Knotenmengen (*node set*) innerhalb von XML-Dokumenten
 - Grundlage für XSLT
 - Operiert auf der logischen Struktur (Baum) des XML-Dokuments
 - Definition von Achsen (*axes*) + Funktionen zur Navigation
 - Knotenarten:
 - RootNode (Wurzelknoten): Nicht Wurzelement sondern dessen „virtueller“ Elternknoten
 - ElementNode (Elementknoten)
 - AttributeNode (Attributknoten), TextNode (Textknoten)
 - NamespaceNode (Namensraumknoten)
 - ProcessingInstructionNode (Verarbeitungsanweisungsknoten)
 - CommentNode (Kommentarknoten)

■ XPath Ausdruck

- Das primäre syntaktische Gebilde in XPath ist ein Ausdruck (Expression), der in einem bestimmten Kontext ausgewertet wird
- Auswertung eines XPath-Ausdrucks liefert eines der folgenden Objekte: Menge von Knoten, Wahrheitswert, Fließkommazahl, Zeichenkette
- Beispiel: „/buchladen/buch[preis>35.00]“

Ausdruck	Aktion im Dokument
knotenname	Selektiert alle Knoten mit Namen "knotenname"
/	Selektiert alle Knoten vom Root aus
//	Selektiert alle Knoten im Dokument, welche der Selektion entsprechen, egal wo sie liegen
.	Selektiert den aktuellen Knoten
..	Selektiert den Elternknoten des aktuellen Knotens
@	Selektiert Attribute

Quelle: [http://www.w3schools.com/xpath/xpath_syntax.asp]

XSL Transformations: Grundlagen

- XSLT = Transformationssprache für XML
 - Beschreibung der Transformation eines XML-Dokumentes in eine andere Struktur
 - Filterung, Sortierung, Nummerierung und ähnliches möglich
 - Steuerung der Transformation durch **unabhängige Regeln** (keine Reihenfolge vorgegeben!)
 - Definition der Regeln über Templates
 - Template definierte die zu selektierende Elemente und die anzuwendenden Aktionen

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  <!-- created 2005-12-12 -->
  <xsl:include href="xslt_stylesheet.xsl" ?>
  <xsl:output method="xml" ?>
  <xsl:template match="/" ?>
    <root>
      <Heuristic: <xsl:value-of select="..." ?>
      <p>The leading manufact
    </root>
  </xsl:template>
</xsl:stylesheet>
```

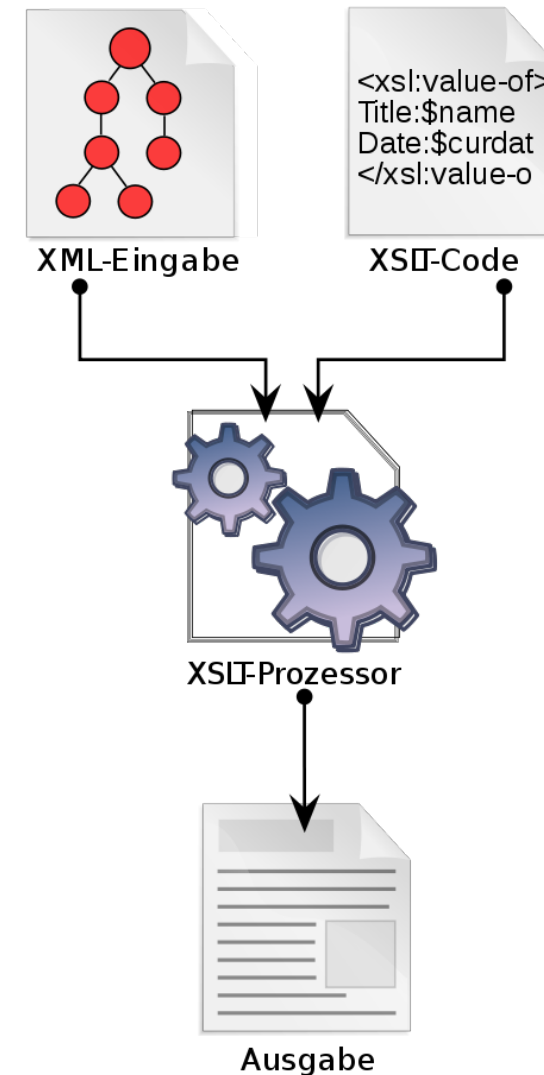
XSL

Quelle: <http://de.wikipedia.org/w/index.php?title=XSLT.svg>

XSL Transformations: Transformationsablauf

■ Ablauf

- XML-Dokument und XSLT-Stylesheet werden vom XSLT Prozessor geladen und verarbeitet
- Prozessor sucht nach passenden Transformationsregeln (Templates) im Stylesheet und wendet diese auf XML an
- Matching über Tag:
`<xsl:template match="...">`
...
`</xsl:template>`



Quelle: [<http://de.wikipedia.org/w/index.php?title=Datei:TempDeXslt015.svg>]

■ Beispiel (XML + XSLT Stylesheet)

```
<catalog>
  <cd>
    <title>Empire
      Burlesque</title>
    <artist>Bob Dylan</artist>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
  </cd>
</catalog>
```

Link zum Beispiel:
[\[http://www.w3schools.com/xsl/tryxslt.Asp?xmlfile=catalog&xsltfile=catalog\]](http://www.w3schools.com/xsl/tryxslt.Asp?xmlfile=catalog&xsltfile=catalog)

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" ... >
  <xsl:template match="/">
    <html> <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr>
          <th>Title</th>
          <th>Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of
              select="title"/></td>
            <td><xsl:value-of
              select="artist"/></td>
          </tr>
        </xsl:for-each>
      </table> </body> </html>
    </xsl:template>
  </xsl:stylesheet>
```

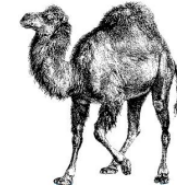
■ Wichtige Tags

- `<value-of>` Liest den Inhalt eines Knotens aus und fügt ihn ins Ausgabedokument ein
- `<for-each>` Selektion und traversieren eines *node sets*
- `<sort>` Sortieren der selektierten Knotenmenge
- `<if>` Einfache konditionale Prüfung
- `<choose>` In Verbindung mit `<xsl:when>` und `<xsl:otherwise>` für multiple konditionale Prüfungen

Einführung PHP: Blick über den Tellerrand

■ Andere Skriptsprachen

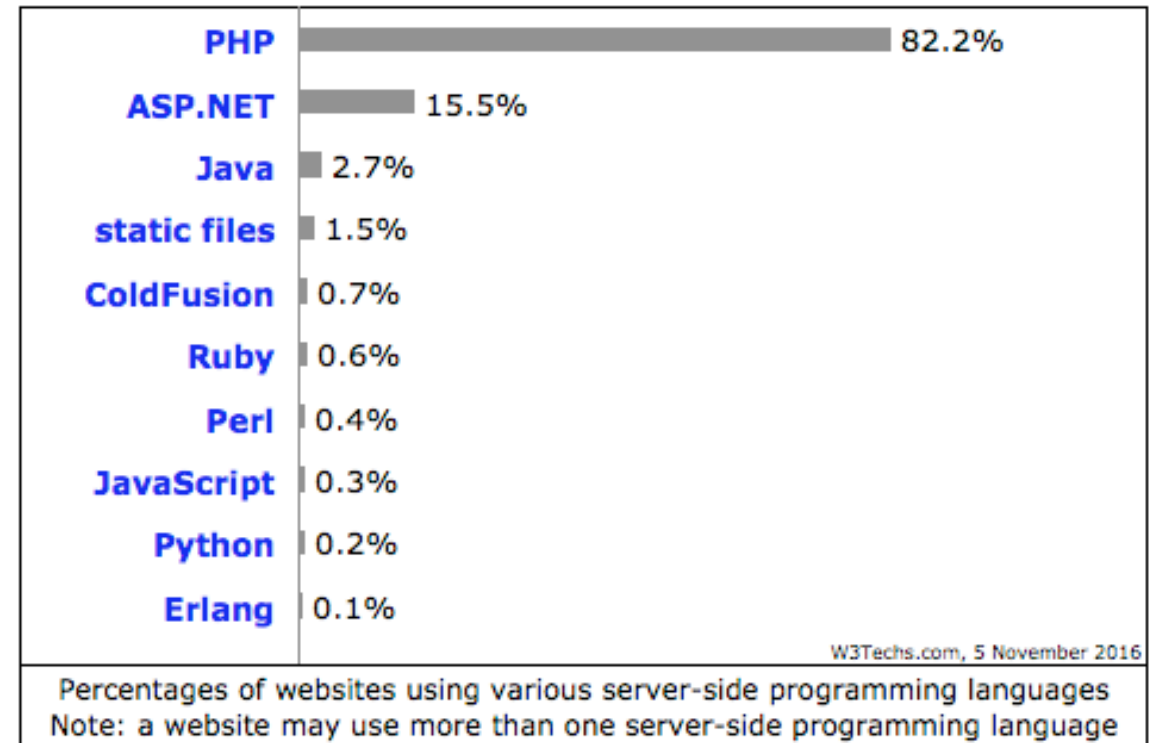
- Perl, 1987
- Python, 1991
- CGI, 1993
- PHP, 1995
- Ruby, 1995
- ASP, 1996
- JSP, 1999



TIOBE Index November 2016,
<http://www.tiobe.com/tiobe-index/>

Pos	Language	Rating
1	Java	18.75%
2	C	9.203%
3	C++	5.515%
4	C#	3.659%
5	Python	3.567%
6	Visual Basic .NET	3.167%
7	PHP	3.125%
8	JavaScript	2.705%
9	Assembly lang.	2.441%
10	Perl	2.361%

Usage of server-side programming languages for websites, W3Techs, November 2016
https://w3techs.com/technologies/overview/programming_language/all



- Gutes Nachschlagewerk: SELFPHP, [\[http://www.selfphp.de/\]](http://www.selfphp.de/)
- „**P**HP: **H**ypertext **P**reprocessor“
 - Serverseitig interpretierte Skriptsprache, Open source
 - Syntax angelehnt an C und Perl
- Einbettung in HTML-Code und serverseitige Ausführung
 - Skripte werden vor Auslieferung interpretiert, übersetzt
- Seit PHP 4 mit Objektkonzept, Objektorientierung
- Breite Unterstützung verschiedener Datenbanksysteme
- Übliche Dateiendungen: **.php** | .php3 | .php4 | ... | .phtml

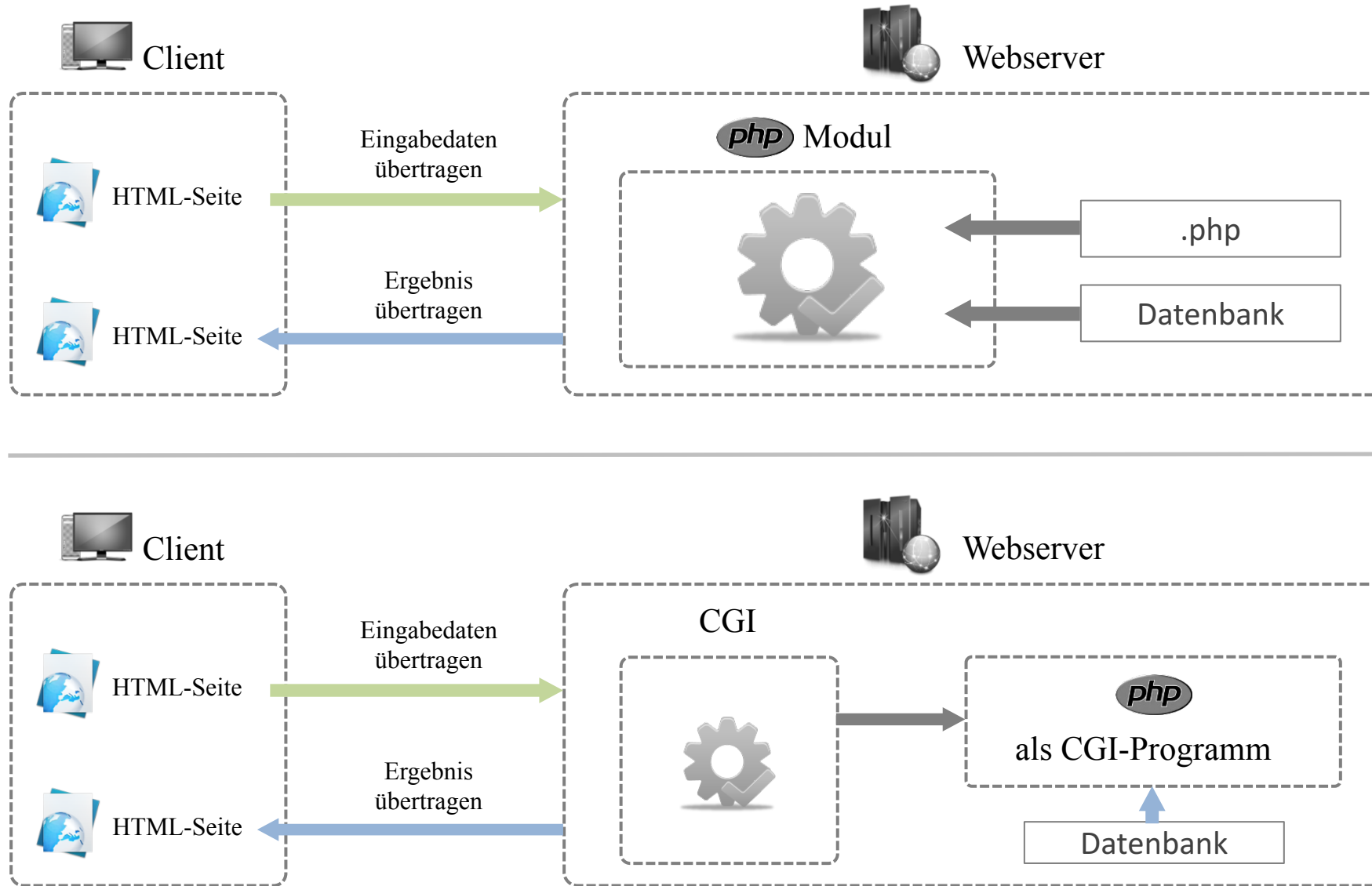


Abbildung nach: <http://selfphp.de/praxisbuch/praxisbuch.php?group=4>

- PHP-Interpreter durchsucht HTML-Code nach Anfangs- und Abschluss-Verarbeitungsinstruktionen `<?php` und `?>`
- PHP-Code Einbettung in HTML-Code
 - XML-Stil (wird am meisten genutzt, ist „sauber“)

```
<?php echo "Einbindung in XML-Stil"; ?>  
<?PHP echo "Einbindung in XML-Stil"; ?>
```

```
<?php  
    echo "Einbindung in XML-Stil";  
?>
```

- Short-Tag, funktioniert nicht immer!

```
<? echo "Einbindung in XML-Stil"; ?>
```

- Javascript-Stil

```
<script language="php">  
    echo "Einbindung im JavaScript-Stil";  
</script>
```

- Minimalbeispiel:

PHP-HTML-Code auf dem Server

```
<html>
  <head>
    <title>KP MI</title>
  </head>
  <body>
    <?php echo "Hello World!"; ?>
  </body>
</html>
```



HTML-Code im Browser (Client)

```
<html>
  <head>
    <title>KP MI</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

■ Allgemeine Syntax

- Ende eines Befehls (o. Befehlszeile) wird mit Semikolon markiert
- Variablennamen beginnen mit \$

```
<?php
    $var = "Hello World";
    echo $var;
?>
```

– Kommentare

```
<?php
    // Alles hinter dieser Markierung ist ein Kommentar
    # Alles hinter dieser Markierung ist ein Kommentar

    /* Alles zwischen dieser Markierung ist ein Kommentar */
    /*
        Alles zwischen dieser Markierung ist ein Kommentar
    */
?>
```

- Allgemeine Syntax
 - Primitive Datentypen: PHP entscheidet zur Laufzeit über Datentyp

```
<?php
    // boolean
    $isPublic = TRUE; // TRUE or FALSE
    $isPublic = 1; // 1 or 0

    // integer, float
    $x = 10;
    $y = 10.123456;

    // string
    $text = "Hello World";

    // array, object
    $num_arr = array( 10, 11, 234, 23 );
    $key_arr = array( "key_1" => 10, "key_2" => TRUE, "key_3" => „Hi" );

    $obj = new ClassName();
?>
```

■ Funktionsaufrufe

- Am Beispiel „Einbindung externer Skripte“

```
<?php  
    include( "datei.inc" );  
?>
```

- Einbindung ermöglicht Verteilung von Code, Programmstruktur
 - Stichwort: zentrale und oft verwendete Funktionen/Klassen
- include() und require() ähnliche Funktion
 - include() wirft „Warning“ bei fehlender Datei
 - require() bricht „Fatal Error“ ab
- Seit PHP 4, include_once() und require_once()
- Benennung der Dateien: Sicherheitsproblem!
 - Auslieferung von Dateien mit bestimmten Dateiendungen

- Definition von Funktionen

- Schlüsselwort „function“

```
<?php
// Definition
function quadratSumme( $value ) {
    $result = $value * $value;
    return $result;
}

// Aufruf: Ergebnis ist 25
echo "Funktion quadratSumme liefert: " . quadratSumme(5);
?>
```

- return-Wert optional, Angabe Datentyp nicht notwendig
- Übergabe von Parametern: „by value“ (Standard) oder „by reference“
(`<?php &$var ?>`)

- Bedingungen
 - if-elseif-Anweisung

```
<?php
    if ($value >= 100) { ... }
    elseif ($value >= 50) { ... }
    else { ... }
?>
```

- Switch-case

```
<?php
    switch ($value) {
        case 100:
            ...
            break;
        case 150:
            ...
            break;
        default:
            ...
            break;
    }
?>
```

■ Schleifen

– while-Schleife

```
<?php
    while () { ... }
?>
```

– do-while-Schleife

```
<?php
    do { ... } while ();
?>
```

– for-Schleife

```
<?php
    for ($i=0;$i<10;$i++) { ... }
?>
```

– foreach-Schleife

```
<?php
    foreach ($array as $value) { ... }
    foreach ($array as $key => $value) { ... }
?>
```


- Auszug PHP-Funktionsumfangs:
 - Array-Funktionen (Sortieren, Suche, ...)
 - Dateisystem-Funktionen (Lesen, Schreiben, Listen, ...)
 - Datums- und Zeit-Funktionen
 - Mail-Funktion
 - Mathematische-Funktionen (Winkel, Runden, ...)
 - MySQL-Funktionen (Verbindung, Schreiben, ...)
 - PDF-Funktionen (Öffnen, Einfügen, Zeichnen, Speichern, ...)
 - Session-Funktionen (Starten, Daten Speichern, ...)
 - String-Funktionen (Suche, Teilen, ...)

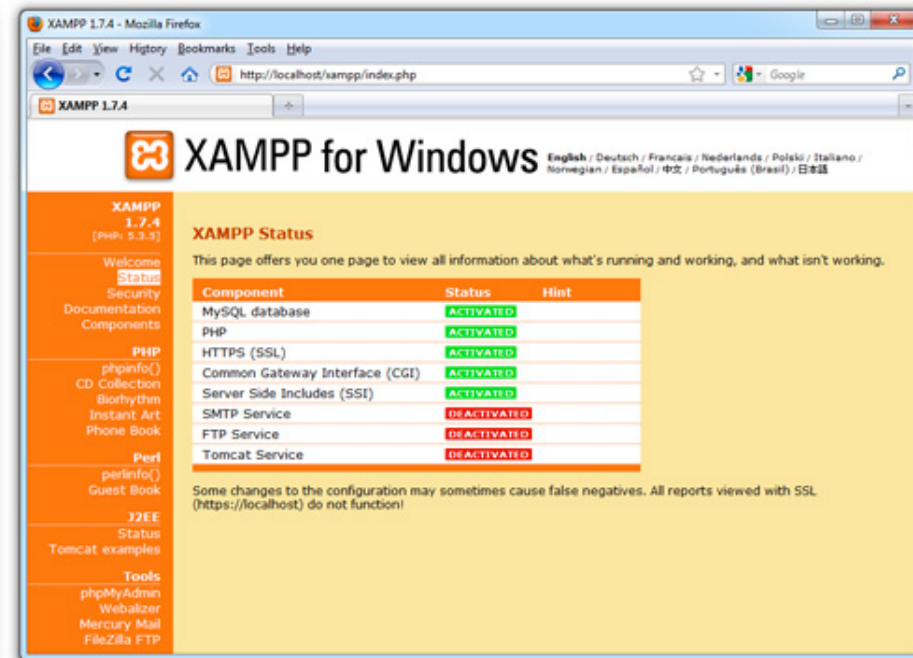
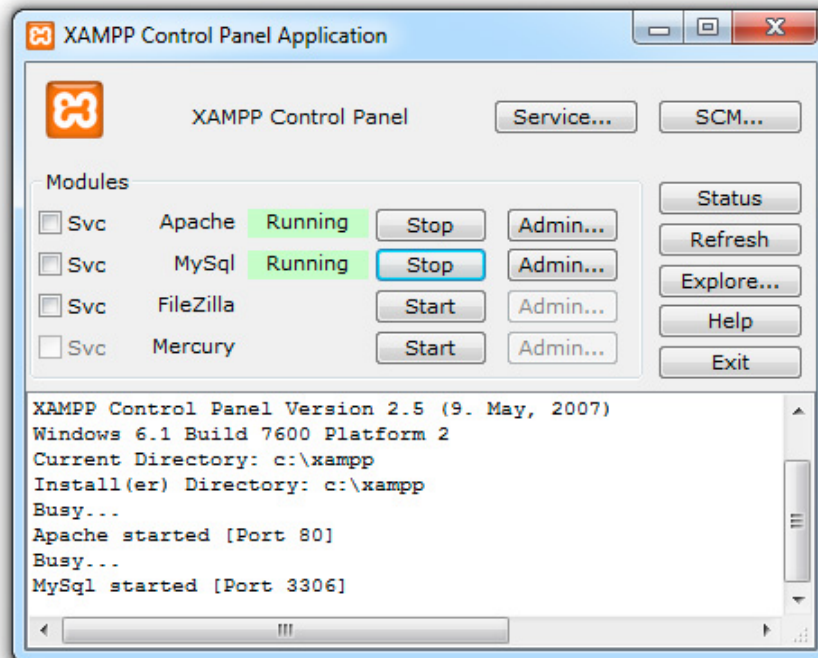
- Gutes Nachschlagewerk: SELFPHP, [\[http://www.selfphp.de/\]](http://www.selfphp.de/)

Teil 3

Hilfreiches, Tipps und Links

Hilfreiches, Tipps und Links

- XAMPP laden und entpacken
 - Website: <https://www.apachefriends.org/de/>
 - Aktuelle Version für alle Systeme (Windows, MacOS X, Linux)
 - XAMPP: **C**ross **A**pache HTTP Server, **M**ySQL, **P**HP and **P**erl
 - Inklusive phpMyAdmin (DB-Verwaltung), FileZilla FTP Server, ...



- XAMPP: Wo lege ich meine Dokumente hin?
 - Ordner im XAMPP-Verzeichnis für alle Web-Dokumente lautet `\htdocs\Team_??\`
 - Bsp. Pfad zur index.html: `127.0.0.1:80\Team_15\index.html`
 - Dort Grundstruktur aus Aufgabe 1 einfügen
- FAQ XAMPP für Windows
 - [\[https://www.apachefriends.org/faq_windows.html\]](https://www.apachefriends.org/faq_windows.html)

Fragen?



Interactive Media Lab Dresden
Professur für Multimedia-Technologie

Kontakt:

Ricardo Langner (ricardo.langner@tu-dresden.de)

Philipp Heisig (philipp.heisig@tu-dresden.de)

Changelog

Datum / Zeit	Beschreibung
2017-11-06 9:00	▪ Initiale Download-Version