WebTech (4)

Prof. Dr.-Ing. S. Gössner

University of Applied Sciences Lippe & Höxter

Inhalt

Was ist ein Server

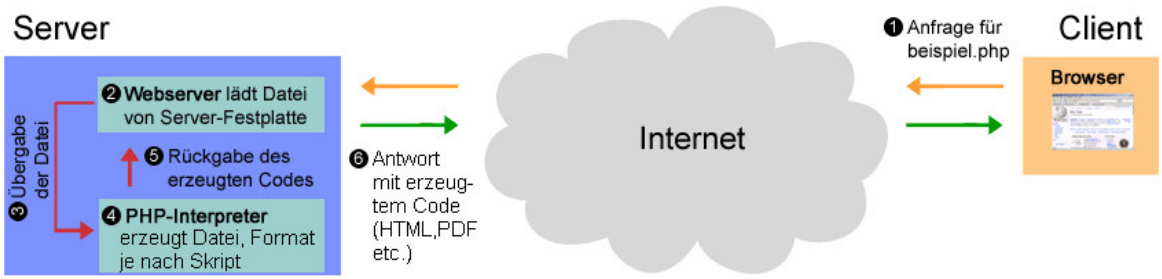
- Ein Server ist ein Programm, welches auf die Kontaktaufnahme eines Client-Programmes wartet und nach Kontaktaufnahme mit diesem Nachrichten austauscht.
- Als Gegenteil zum *Client/Server-Prinzip* wird das *Peer-To-Peer-Prinzip (P2P)* aufgefasst.
- Webserver sind Serverdienste, die das HTTP-Protokoll auf einem Rechner zur Verfügung stellen.
- Weitere Server:
  - Datenbankserver
  - Print-Server
  - Root-Server
  - Proxy-Server
  - Spieleserver

PHP - Beispiel einer serverseitigen Programmiersprache

PHP ...

- heisst ***P****HP* ***H****ypertext* ***P****reprozessor (rekursives Akronym)* ehemals ***P****ersonal* ***H****ome* ***P****age Tools*.
- ist eine Skriptsprache mit einer an C angelehnten Syntax.
- wurde 1995 von Rasmus Lerdorf definiert und implementiert.
- wird heute von Zend Technologies Ltd. weiterentwickelt.
- ist wegen seiner Einfachheit sehr populär.
- ist etwas unkoordiniert gewachsen.
- wartet in seiner aktuellen Version 5.0 mit objektorientierten Eigenschaften auf.

PHP - Funktionsweise



1. Webclient startet Anfrage *(HTTP-Request)* an Webserver nach einer PHP-Datei.
2. Der Server lädt die PHP-Datei von seiner Festplatte und …
3. übergibt die PHP-Datei an den PHP-Interpreter.
4. Der PHP-Interpreter arbeitet das/die PHP-Skript(e) ab.
5. Der Interpreter gibt das Resultat *(HTML-Datei)* an den Server zurück.
6. Der Webserver liefert das Ergebnis an den Webclient aus.

[Quelle: Wikipedia]

Übertragung von Formulardaten

Wir wollen in einem Beispiel die Email Adresse einer Studierenden übertragen.

Html Datei: *student01.html*

```
<html>
  <head>
     <title>Eingabe und Übertragung von Formulardaten</title>
  </head>
  <body>
    <h1>Studentenerfassung</h1>
    <form method="post" action="student01.php">
      Email:
      <input type="text" id="email" name="email" />
      <input type="submit" value="senden .." />
    </form>
  </body>
</html>
```

Php Datei: *student01.php*

```
<?php
print("Email an Server angekommen: " . $_POST["email"]);
?>
```

[Ausführen]

Übertragung von Formulardaten (2)

In Erweiterung des Beispiels wollen wir die Eingabedaten serverseitig validieren.

Html Datei: *student02.html*

```
<html>
  <head>
     <title>Eingabe und Übertragung von Formulardaten</title>
  </head>
  <body>
    <h1>Studentenerfassung</h1>
    <form method="post" action="student02.php">
      Email:
      <input type="text" id="email" name="email" />
      <input type="submit" value="senden .." />
    </form>
  </body>
</html>
```

Die Plausibilitätsprüfung soll sehr einfach sein und überprüft daher lediglich die Existenz eine @ innerhalb der Email Adresse.

Php Datei: *student02.php*

```
<?php
if (preg_match("/@/", $_POST["email"]) > 0)
   print("valid email: " . $_POST["email"]);
else
   print("invalid email: " . $_POST["email"]);
?>
```

[Ausführen]

Übertragung von Formulardaten (3)

Nun soll der Unterschied zur clientseitigen Validierung demonstriert werden.

Html Datei: *student03.html*

```
<html>
  <head>
     <title>Eingabe und Übertragung von Formulardaten</title>
     <script type="text/javascript">
        function checkEmail() {
           var emailbox = document.getElementById("email");
           if (emailbox && emailbox.value.match(/@/) != null) // ok ..
              return true;
           else {
              emailbox.style.backgroundColor = "yellow";
              alert("\"" + emailbox.value + "\" ist keine gescheite Emailadresse!");
           }
           return false;
        }
     </script>
  </head>
  <body>
    <h1>Studentenerfassung</h1>
    <form method="post" action="student03.php" onsubmit="return checkEmail();">
      Email:
      <input type="text" id="email" name="email"
             onfocus="this.style.backgroundColor='white'" />
      <input type="submit" value="senden .." />
    </form>
  </body>
</html>
```

Die Serverseite bleibt unverändert. Die Redundanz der Validierung ist erwünscht.

Php Datei: *student03.php*

```
<?php
if (preg_match("/@/", $_POST["email"]) > 0)
   print("valid email: " . $_POST["email"]);
else
   print("invalid email: " . $_POST["email"]);
?>
```

[Ausführen]

Übertragung von Formulardaten (4)

Die Verarbeitung der Eingabedaten beinhaltet im Allgemeinen ein Speichern für späteren Zugriff.

Html Datei: *student04.html*

Die Clientseite bleibt unverändert gegenüber *student03.html*.

Php Datei: *student04.php*

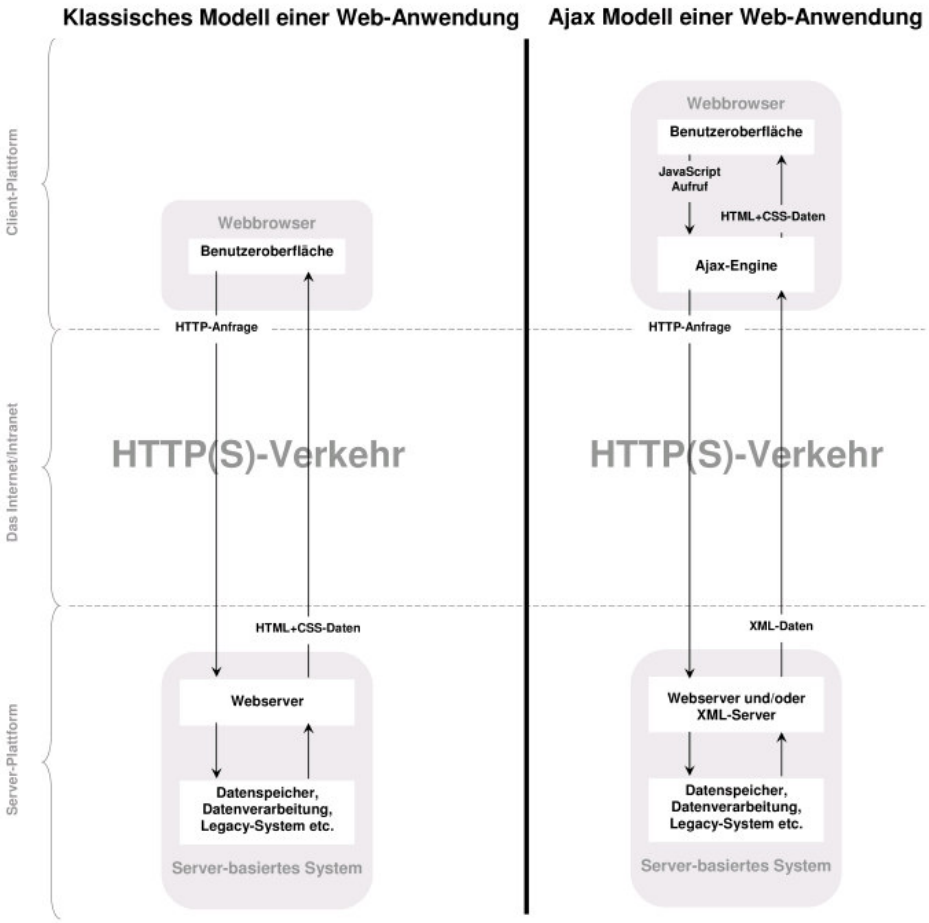
```
<?php
function loadFile($uri) {
   $content = "";
   $file = ($uri != "") ? fopen($uri, "rb") : null;
   if ($file != null) {
      while (!feof($file))
         $content .= fread($file, 100);
      fclose($file);
   }
   else
      printf("Cannot read file '%s'", $uri);
   return $content;
}
function saveFile($uri, $src) {
   if ($file = fopen($uri, "w")) {
      if (!fwrite($file, $src))
         printf("Cannot write to file '%s'", $uri);
      fclose($file);
   }
   else
      printf("Cannot open file '%s'", $uri);
}

if (preg_match("/@/", $_POST["email"]) > 0) {
   $emails = loadFile("students.eml");
   $emails .= "\n" . $_POST["email"];
   saveFile("students.eml", $emails);
}
else
   print("invalid email: " . $_POST["email"]);
?>
```

[Ausführen]

moderne Webanwendung - asynchrone Kommunikation

- Konventionelle Webanwendungen bedienen sich der Formulardatenübertragung.
  - Als Vorteil ist die breite Browserunterstützung und das umfangreiche serverseitige Softwarefundament zu nennen.
  - Als Nachteil wird die starre, an Formularelemente gebundene Kommunikation empfunden:
    1. Benutzer gibt Daten in Felder ein.
    2. Benutzer sendet vollständige Daten an Server
    3. Server sendet neues Dokument *(neue Webseite)* zurück.
- Moderne Webanwendungen benutzen ein *HttpRequest*-Objekt des Browsers zur asynchronen Kommunikation zwischen *Client* und *Server*.
- Die Kombination *Asynchrones Javascript + HttpRequest + XML (Ajax)* wird als leistungsfähige Basis für flexible Webanwendungen angesehen.

Ajax


Klassisches Modell einer Web-Anwendung
Ajax Modell einer Web-Anwendung
Client-Plattform
Das Internet/Intranet
Server-Plattform
Webbrowser
Benutzeroberfläche
HTTP-Anfrage
HTTP(S)-Verkehr
HTML+CSS-Daten
Webserver
Datenspeicher, Datenverarbeitung, Legacy-System etc.
Server-basiertes System
Webbrowser
Benutzeroberfläche
JavaScript Aufruf
HTML+CSS-Daten
Ajax-Engine
HTTP-Anfrage
HTTP(S)-Verkehr
XML-Daten
Webserver und/oder XML-Server
Datenspeicher, Datenverarbeitung, Legacy-System etc.
Server-basiertes System

Ajax - Beispiel

In einem Vorlesungsbeispiel wollen wir drei quadratische Felder mit Namen belegen. Diese Namen sollen gleichzeitig in einer serverseitigen XML Datei gesichert werden.

HTML Datei

```
<html>
  <head>
    <title>Boxes</title>
    <style type="text/css">
       div.box { width:190px; height:190px; background-color: lightblue; padding: 3px;
                 border:1px solid black; font-size:150%; text-align: center; vertical-align: middle; }
    </style>
  </head>
  <body>
    <h1> Boxes </h1>
    <div id="b1" class="box" style="position:absolute; left:100px; top:100px;"> n/a </div>
    <div id="b2" class="box" style="position:absolute; left:300px; top:100px;"> n/a </div>
    <div id="b3" class="box" style="position:absolute; left:500px; top:100px;"> n/a </div>
  </body>
</html>
```

[Ansicht]

XML Datei

```
<?xml version="1.0"?>
<boxes>
  <box id="b1" name="n/a" />
  <box id="b2" name="n/a" />
  <box id="b3" name="n/a" />
</boxes>
```

Ajax - Beispiel (2)

Die Anforderungen (clientseitig):

1. Beim Anklicken einer Box wird
   a. bei freier Box *(n/a)* ein Textfeld und ein *Ok*-Button sichtbar.
   b. bei reservierter Box eine Fehlermeldung ausgegeben.
2. Bei Betätigung des *Ok*-Button wird Textfeld und Button entfernt und nunmehr der Inhalt des Textfeldes in die Box eingetragen.

HTML Datei

```
<html>
  <head>
    <title>Boxes</title>
     <script type="text/javascript">
        function setBox(box) {
           if (box.innerHTML == " n/a ") { // free box ..
              box.onclick = null;
              box.innerHTML = '<input id="name" style="width:80px;" type="text" size="90px" value="" /><br/>'+
                              '<button onclick="reserveBox(this.parentNode, document.getElementById(\'name\').value);">ok</
           }
           else
              alert("Box ist leider schon belegt!");
        }
        function reserveBox(box, name) {
           box.innerHTML = name;
           box.onclick = setBox;
        }
     </script>
     <style type="text/css">
       div.box { width:190px; height:190px; background-color: lightblue; padding: 3px;
                 border:1px solid black; font-size:150%; text-align: center; vertical-align: middle; }
     </style>
  </head>
  <body>
    <h1> Boxes </h1>
    <div id="b1" class="box" style="position:absolute; left:100px; top:100px;" onclick="setBox(this);"> n/a </div>
    <div id="b2" class="box" style="position:absolute; left:300px; top:100px;" onclick="setBox(this);"> n/a </div>
    <div id="b3" class="box" style="position:absolute; left:500px; top:100px;" onclick="setBox(this);"> n/a </div>
  </body>
</html>
```

[Ansicht]

Ajax - Beispiel (3)

Nun soll die Namensangabe mittels eines asynchronen Http-Request an den Webserver übertragen werden, mit der Aufforderung:

1. Den Namen zu validieren.
2. Den Namen,
   a. wenn ok, in die XML Datei einzutragen.
   b. wenn nicht ok, eine Fehlermeldung zu generieren.
3. *ok* oder Fehlermeldung an Client zurückliefern.
4. Den Namen in die Box eintragen (wenn *ok*) oder den ursprünglichen Inhalt wieder herstellen (*n/a*).

Ajax - Beispiel (4)

Clientseite

```
<html>
  <head>
    <title>Boxes</title>
     <script type="text/javascript" src="/js/http.js"> </script>
     <script type="text/javascript">
        function setBox(e) {
           var box = e.target;
           if (box.innerHTML == "n/a") { // free box ..
              box.onclick = null;
              box.innerHTML = '<input id="name" style="width:80px;" type="text" size="90px" value="" /><br/>'+
                              '<button onclick="reserveBox(this.parentNode, document.getElementById(\'name\').value);">ok</
           }
           else
              alert("Box ist leider schon belegt mit " + box.innerHTML);
        }
        function reserveBox(box, name) {
           var callback = function(msg) {
              if (msg == "ok")
                 box.innerHTML = name;
              else {
                 box.innerHTML = "n/a";
                 alert(msg);
              }
              box.onclick = setBox;
           };
                           Http.post("reserve=name&boxid="+box.id+"&name="+name, "boxes.php", callback);
        }
     </script>
     <style type="text/css">
       div.box { width:190px; height:190px; background-color: lightblue; padding: 3px;
                 border:1px solid black; font-size:150%; text-align: center; vertical-align: middle; }
     </style>
  </head>
  <body>
    <h1> Boxes </h1>
    <div id="b1" class="box" style="position:absolute; left:100px; top:100px;" onclick="setBox(event);">n/a</div>
    <div id="b2" class="box" style="position:absolute; left:300px; top:100px;" onclick="setBox(event);">n/a</div>
    <div id="b3" class="box" style="position:absolute; left:500px; top:100px;" onclick="setBox(event);">n/a</div>
  </body>
</html>
```


Ajax - Beispiel (5)

Serverseite

```
<?php
   require_once("XPath.class.php");
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      if (isset($_POST["reserve"],$_POST["boxid"],$_POST["name"])) {
         $xpath = new XPath();
         $xpath->importFromFile("boxes.xml");
         if (count($xpath->match("//box[@id='".$_POST["boxid"]."']"))) {  // valid boxid ..
            $name = $xpath->getAttributes("//box[@id='".$_POST["boxid"]."']", "name");
            if ($name == "n/a") {  // still not reserved ..
               if (count($xpath->match("//box[@name='".$_POST["name"]."']")) == 0) { // name not found elsewhere ..
                  $xpath->setAttribute("//box[@id='".$_POST["boxid"]."']", "name", $_POST["name"]);
                  $xpath->exportToFile("boxes.xml");
                  print("ok");
               }
               else
                  print($_POST["name"] . " kann nicht zweimal belegen!");
            }
            else
               print("Box " . $_POST["boxid"] . " bereits belegt mit " . $name);
         }
         else
            print("Unbekannte Box: ".$_POST["boxid"]);
      }
   }
?>
```

[Ausführen]