PHP Entwicklung

SELFPHP
http://www.selfphp.info

- Eigene Funktionen definieren mit *function*

```
function hello($text1, $text2)
{
    echo "Hello World <br>";
    echo "$text1 $text2 <br>";
    return true; // Rückgabewert
}
```

- Optionale Funktions-Parameter mit Default-Wert:

```
function hello($text1, $text2="Test")
```

- Während der Entwicklung: alle Fehlercodes ausgeben mit

```
error_reporting(E_ALL);
```

Exceptions

erst ab PHP5
- wie bei Java, C++…

```
<?php
try {
    $error = 'Always throw this error';
    throw new Exception($error);
    // Code following an exception is not executed.
    echo 'Never executed';
}
catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}
// Continue execution
echo 'Hello World';
?>
```

Strings

Ein String kann auf drei verschiedene Weisen geschrieben werden.

- Einfache Anführungszeichen (single quote) `" als "`

```
echo 'Variablen werden $nicht $ausgewertet\n';
//Ausgabe: Variablen werden $nicht $ausgewertet\n
```

- Doppelte Anführungszeichen (double quote) `" als &quot; oder \"`

```
$myvar = 'Variable';
echo "${myvar}n werden ausgewertet\n";
//Ausgabe: Variablen werden ausgewertet [newline]
```

komplexe Berechnungen und Abgrenzung von Variablennamen in {}

- Heredoc Notation

  Stringbehandlung wie mit doppelten Anführungszeichen - nur ohne Anführungszeichen; Zeilenumbrüche werden übernommen

  `" als "`

```
echo <<<EOT
${myvar}n werden ausgewertet
EOT;
//Ausgabe: Variablen werden ausgewertet [newline]
```

kein Semikolon!

Schluss"tag" muss in der ersten Spalte stehen!

Sonderzeichen in Strings

- im erzeugten HTML bzw. den ECMA-Script Anteilen muss die jeweilige Syntax beachtet werden
- das Stringformat erfordert die Ersetzung mancher Sonderzeichen
  - ⇨ Stringbegrenzer ist `"` und muss zur Ausgabe "escaped" werden: `\"`
  - ⇨ für ECMA-Script sogar "doppelt": `&quot;`
- für die Weiterverarbeitung und Speicherung (z.B. in der Datenbank) sollen Daten in "reiner Form" stehen

⇨ PHP bietet diverse Funktionen zum Wandeln von Strings zwischen den Formaten

Sonderzeichen in Strings - sprachübergreifend

- übermittelte Formulardaten werden Strings für
  - ⇨ HTML zum Anzeigen (ersetzt & < > " ')

```
$Wert = htmlspecialchars($Wert);
```

  - ⇨ SQL für Anfragen etc. (ersetzt " ')

```
$Wert = mysql_real_escape_string($Wert);
```

`htmlentities()` ersetzt auch die Umlaute!

- Erzeugen von JavaScript in HTML aus PHP-echo:

PHP ⬇

Browser ⬇

```
echo("<p onclick=\"alert(&quot;Hallo&quot;);\">");
      <p onclick="alert(&quot;Hallo&quot;);">
                  alert("Hallo");
```

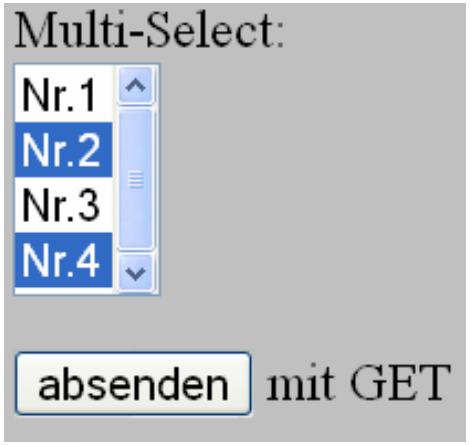
- Einfacher mit heredoc-Notation:

```
<?php
echo <<<EOT
<!– Hier steht der ganz normale HTML-Code -->
<!DOCTYPE…
</html>
EOT;
?>
```

Übergabe mit Mehrfachauswahl (I)

```
<form action="21_FormularEcho.php" method="get">
    <p>Multi-Select:<br>
    <select name="myselect" size="4" multiple>
        <option value="1">Nr.1</option>
        <option value="2">Nr.2</option>
        <option value="3">Nr.3</option>
        <option value="4">Nr.4</option>
    </select>
    </p>
    <p><input type="submit" value="absenden"> mit GET</p>
</form>
```

HTML



mit POST sieht man nichts, aber das Problem bleibt!

überträgt: myselect=2&myselect=4

PHP liefert: myselect=4 (als Umg.variable in $_GET)

PHP

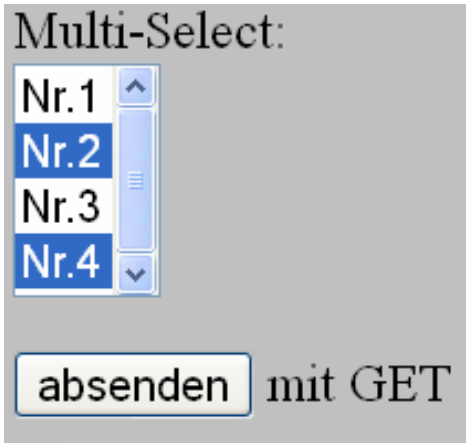
⇨ Es wird nur der letzte Wert als Umgebungsvariable übernommen!

Übergabe mit Mehrfachauswahl (II)

```
<form action="21_FormularEcho.php" method="get">
    <p>Multi-Select:<br>
    <select name="myselect[]" size="4" multiple>
        <option value="1">Nr.1</option>
        <option value="2">Nr.2</option>
        <option value="3">Nr.3</option>
        <option value="4">Nr.4</option>
    </select>
    </p>
    <p><input type="submit" value="absenden"> mit GET</p>
</form>
```

array

[ ]



überträgt: myselect%5B%5D=2&myselect%5B%5D=4

PHP `print_r($_GET);` liefert jetzt:

Befehl zum Ausgeben von Arrays

Array ([myselect] => Array ([0] => 2 [1] => 4)) myselect=Array

Zugriff auf Position `$i` über

```
$MYS = $_GET[myselect];
echo ($MYS[$i]);
```

Formulare und Sicherheit

später ausführlicher

- wer weiß schon, welche Daten ein gehacktes Formular übermittelt ...
- deshalb nicht die übermittelten Parameter auswerten:

```
foreach ($_POST as $Name => $Wert)
   $Datensatz[$Name] = $Wert;
```

- sondern die erwarteten Parameter:

```
if (isset ($_POST["ElemA"]) &&
    istZahl($_POST["ElemA"])
   $Datensatz["ElemA"] = $_POST["ElemA"];
Speichern ($Datensatz);
```

Syntax und Wertebereich überprüfen

Überprüfung auf unerwünschte Zeichen

- am einfachsten mit regulären Ausdrücken

```
bool preg_match (string pattern, string subject);
```

Beispiele

- Dezimalzahl mit Vorzeichen und max. 10 Ziffern:

```
$isDecimal = preg_match("/-{0,1}[0-9]{1,10}/", $p);
```

  ⇨ Minus darf 0 oder 1-mal auftreten, dann

  1 bis 10 Zeichen aus der Menge `0-9`

- übliche Bezeichner-Syntax:

```
$isAlphaNum = preg_match(
                    "/^[a-zA-Z][a-zA-Z0-9_]+$/", $p);
```

  ⇨ zuerst ein Buchstabe und dann Buchstaben, Zahlen oder _

Vorzeitiger Abbruch

- normalerweise endet ein PHP-Programm am Dateiende
  - ⇨ vorzeitiger Abbruch bei Fehlern

- Abbruch mit Text-Meldung an Browser

```
die ("ungültiger Parameter");
```

- Abbruch mit Fehlercode

```
exit (5);
```

Immer an die HTML-Schachtelungsstruktur denken, damit die Meldung ordentlich formatiert wird!

Objektorientierung

- Klassen und Objekte ähnlich Java
  - ⇨ Objekte werden ausschließlich mit `new` erzeugt und über Referenzen angesprochen
  - ⇨ Attribute deklarieren mit: `var $Attribut;`
  - ⇨ Basisklassenkonstruktor explizit aufrufen

    `parent::__construct()`

    `parent::` bezeichnet Basisklasse

    `$this` verweist auf Objekt

    `->` für Zugriff auf Attribute (ohne `$`) und Methoden
- keine Mehrfachvererbung
- Interfaces können definiert werden
- wenig genutzt in Bibliotheken
  - ⇨ Bibliotheken bestehen oft aus Gruppen von Funktionen
  - ⇨ Musterbeispiel: herstellerunabhängige Kapselung des Datenbankzugriffs in PEAR::DB

Plattformunabhängigkeit ?

- ja: derselbe Interpreter für viele Betriebssysteme
- aber: jede Installation ist anders
  - ⇨ sehr viele Features abhängig von Konfigurationsparametern in PHP.INI
  - ⇨ und / oder auch von Compiler-Schaltern beim Build
  - ⇨ Abfragefunktionen in der Gruppe "PHP options & information"

besser wäre: in der Anwendung

- kein Problem, wenn der Interpreter nur eine Anwendung bedient
  - ⇨ dann kann man ihn passend konfigurieren (muss natürlich bei PHP-Update bewahrt werden)
  - ⇨ aber sehr nachteilig bei mehreren Anwendungen
- interessanter Weg zur Behinderung von Portabilität !

Zusammenfassung

- Grundlagen
  - ⇨ Idee
  - ⇨ Historie und Verbreitung
- Notation
  - ⇨ Grundstil C++
  - ⇨ Integration in HTML `<?php ... ?>`
  - ⇨ Variablennamen beginnen mit `$`
  - ⇨ Strings `.`, `"..."`, `'...'`, `echo <<<EOT`
  - ⇨ Arrays und deren Übergabe
  - ⇨ Umwandlung von Sonderzeichen
  - ⇨ Objektorientierung

Jetzt können wir mächtige Skripts schreiben und laufen lassen
– aber was tun wir damit?

Entwicklung webbasierter Anwendungen

10. Kapitel: Datenbankanbindung mit PHP

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Webserver und Datenbanken

- man *könnte* Daten server-seitig in Dateien ablegen, besser ist aber fast immer eine Datenbank
  - ⇨ Shop- und Buchungssysteme, Content Management Systeme
- PHP bietet einfache Schnittstellen zu vielen Datenbanken
  - ⇨ besonders beliebt, weil kostenlos: MySQL und PostgreSQL
- häufig im Bundle:

  Linux bzw. Windows + Apache + MySQL + PHP

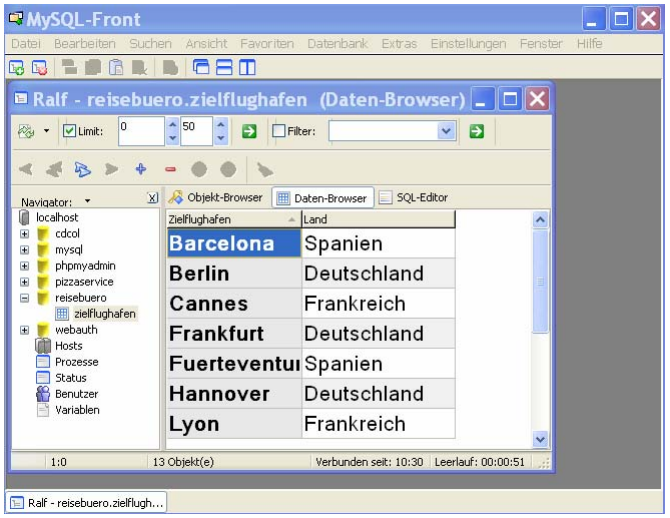
  z.B. auch bei XAMPP

  LAMP / WAMP

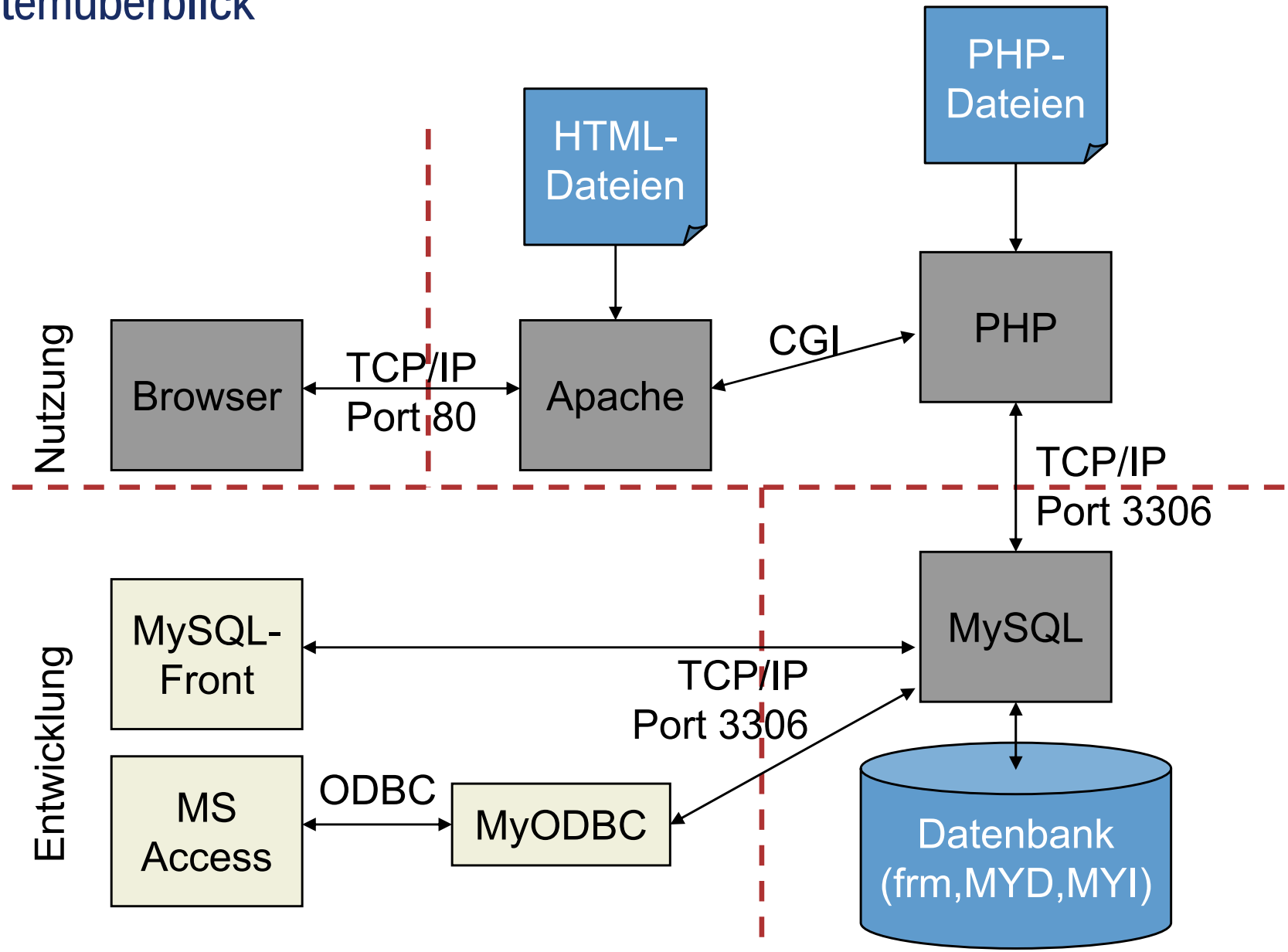
  - ⇨ in manchen Versionen von XAMPP (z.B. V 1.4.13) ist die PHP-Anbindung an MySQL nicht mehr automatisch eingebunden
  - ⇨ in PHP.ini aktivieren: `extension=php_mysql.dll` (unter Windows)

MySQL

- MySQL
  - ⇨ Hauptnachteil: keine Unterstützung von Transaktionen mit MyISAM
    - wohl aber mit InnoDB- oder BerkeleyDB-Tabellen
  - ⇨ PHP selbst benötigt nur den Datenbank-Server mysqld.exe
  - ⇨ Front-End ist erforderlich für DB-Entwicklung und -Administration
    - schön und kostenlos, aber nur für Windows:
      Ansgar Becker's **MySQL-Front** (http://www.mysqlfront.de/)
    - sehr komfortabel durch QBE
      (Query by Example):
    - Microsoft Access über **MyODBC**
    - oder auch PHP-Skripte **phpMyAdmin**
      (http://localhost/phpmyadmin
      - o auch Remote verwendbar
      - o wird mit XAMPP installiert



Systemüberblick



SQL - zur Erinnerung

- ganz primitiv: MySQL über die Kommandozeile
  - ⇨ mysql.exe ist das mitgelieferte Kommandozeilen-Front-End

- \mysql\bin\mysql.exe bzw. C:\Programme\.....\xampp\mysql\bin\mysql.exe
  - ⇨ use Reisebuero

```
SELECT Zimmerart, Kategorie FROM hotel;
UPDATE hotel SET Preis="150" WHERE Preis="116";
DELETE FROM hotel WHERE Ort="Alanya";
INSERT INTO zielflughafen SET Zielflughafen="Rom", Land="Italien";
```

  - ⇨ quit

MySQL-Schnittstelle in PHP

für jede DB eine andere Funktionsbibliothek ☹

- TCP/IP-Verbindung aufbauen

```
$Connection = mysql_connect("localhost:3306");
```

  - ⇨ user und password könnten noch angegeben werden
- Datenbank auswählen (mysqld verwaltet evtl. mehrere)

```
mysql_select_db ("Reisebuero", $Connection);
```

- Ergebnistabelle abfragen oder SQL-Aktion ausführen

```
$Recordset = mysql_query ($Abfrage, $Connection);
```

- nächste Zeile aus Ergebnistabelle in Array übertragen

```
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
```

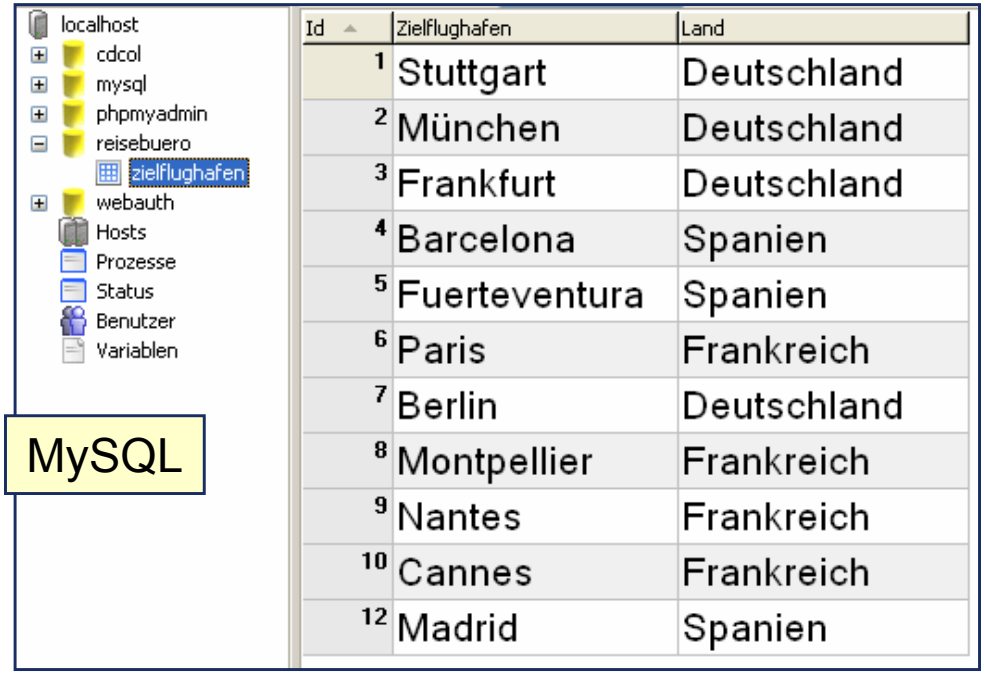
- alle Felder aller Datensätze abarbeiten

```
foreach($Record as $Name => $Wert) { ...}
```

- TCP/IP-Verbindung schliessen

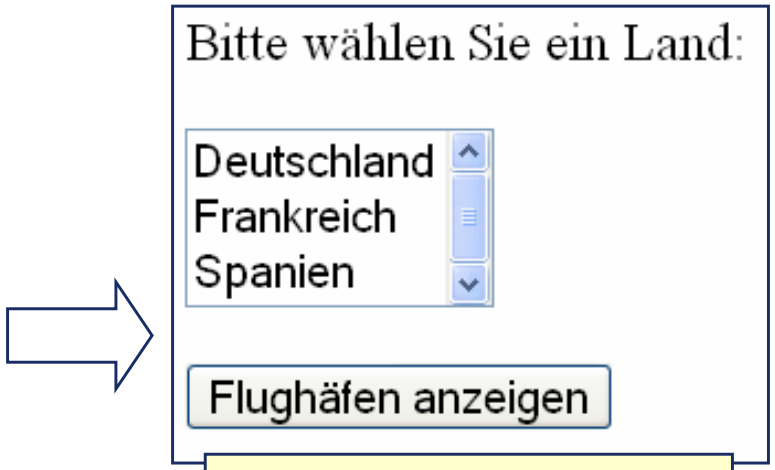
```
mysql_close($Connection);
```

Anwendungsbeispiel



MySQL

Id	Zielflughafen	Land
1	Stuttgart	Deutschland
2	München	Deutschland
3	Frankfurt	Deutschland
4	Barcelona	Spanien
5	Fuerteventura	Spanien
6	Paris	Frankreich
7	Berlin	Deutschland
8	Montpellier	Frankreich
9	Nantes	Frankreich
10	Cannes	Frankreich
12	Madrid	Spanien



generierte Auswahlliste

generierte Ergebnistabelle

Id	Zielflughafen	Land	Zielflughafen(Land)
6	Paris	Frankreich	Paris(Frankreich)
8	Montpellier	Frankreich	Montpellier(Frankreich)
9	Nantes	Frankreich	Nantes(Frankreich)
10	Cannes	Frankreich	Cannes(Frankreich)

Beispiel: Verbindungsaufbau

```php
<?php
// MIME-Type der Antwort definieren (*vor* allem HTML):
header ("Content-type: text/html");
// alle möglichen Fehlermeldungen aktivieren:
error_reporting (E_ALL);

// SQL-Abfrage festlegen:
$SQLabfrage = "SELECT Land FROM zielflughafen GROUP BY Land";

// Datenbank öffnen und abfragen:
$Connection = mysql_connect("localhost", "", "");
if (!$Connection)
    die ("Could not connect");
if (!mysql_select_db ("Reisebuero", $Connection))
    die ("Could not select database");
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Query failed");
?>
```

Zugang ohne User und Passwort – nur für Tests!

Beispiel: Generierung einer Auswahlliste

```
<body>
    <p>Bitte wählen Sie ein Land:</p>
    <form name="Auswahl" action="31_SQL_Ergebnis.php" method="POST">
<?php
$AnzahlRecords = mysql_num_rows ($Recordset);
echo ("\t\t<p><select name=\"AuswahlLand\" size=\"$AnzahlRecords\">\n");
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
while ($Record) {
    $Land = $Record["Land"];
    echo ("\t\t\t<option>$Land</option>\n");
    $Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
}
echo ("\t\t</select></p>\n");

mysql_close($Connection);
?>
        <p>
        <input type="submit" value="Flughäfen anzeigen">
        </p>
    </form>
</body>
```

Beispiel: gefilterte Datenbankabfrage

```php
// SQL-Abfrage aus Formulardaten bestimmen:
$Auswahl = "";
if (isset($_POST["AuswahlLand"]))
    $Auswahl = "WHERE Land = \"".$_POST["AuswahlLand"]."\"";
$SQLabfrage = "SELECT * FROM zielflughafen ".$Auswahl;

// Datenbank öffnen und abfragen:
$Connection = mysql_connect("localhost", "", "");
if (!$Connection)
    die ("Could not connect");
if (!mysql_select_db ("Reisebuero", $Connection))
    die ("Could not select database");
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Query failed");
?>
```

Beispiel: Abfrageergebnis als Tabelle anzeigen

```php
<?php
// generiert die HTML-Tabelle mit Zielflughäfen:
$ersteZeile = true;
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
while ($Record) {
    if ($ersteZeile) {
        $ersteZeile = false;
            echo "\t\t<tr>\n";
            foreach($Record as $Name => $Wert) {
            $Name = htmlspecialchars($Name, ENT_QUOTES);
            echo "\t\t\t<th>$Name</th>\n";
        }
        echo "\t\t\t<th>Zielflughafen(Land)</th>\n";
        echo "\t\t</tr>\n";
    }
    $Land = htmlspecialchars($Record["Land"], ENT_QUOTES);
    $Zielflughafen = htmlspecialchars($Record["Zielflughafen"], ENT_QUOTES);
    echo "\t\t<tr>\n";
    echo "\t\t\t<td>$Zielflughafen</td>\n";
    echo "\t\t\t<td>$Land</td>\n";
    echo "\t\t\t<td>$Zielflughafen($Land)</td>\n";
    echo "\t\t</tr>\n";
    $Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
}
?>
```

Beispiel: Datensatz hinzufügen

```
// Doppeleintrag verhindern:
$SQLabfrage = "SELECT * FROM zielflughafen WHERE ".
    "Zielflughafen = \"$eZielflughafen\" AND Land = \"$eLand\"";
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Abfrage fehlgeschlagen: ".$SQLabfrage);

if (mysql_num_rows($Recordset)>0)
    $Fehlermeldung = "Dieser Flughafen ist bereits eingetragen.";
else {
    $SQLabfrage = "INSERT INTO zielflughafen SET ".
        "Zielflughafen = \"$eZielflughafen\", Land = \"$eLand\"";
    $Recordset = mysql_query ($SQLabfrage, $Connection);
    if (!$Recordset)
        die ("Abfrage fehlgeschlagen: ".$SQLabfrage);
    $eZielflughafen = "";
    $eLand = "";
}
```

Beispiel: DB gekapselt in Klasse

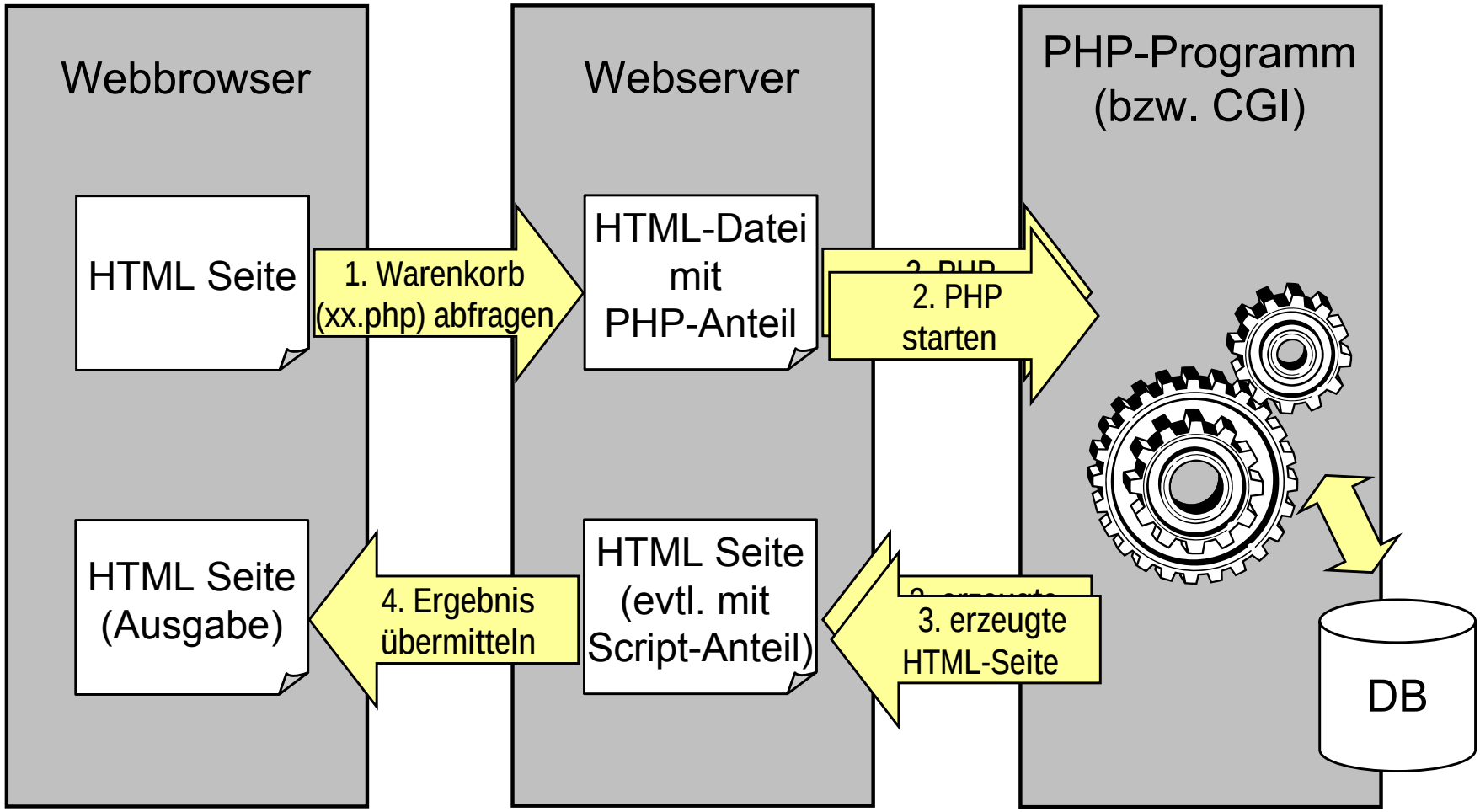
```
class CDatabase {
    var $Connection;
    var $Recordset;

    //*****************************************************
    function Execute ($AktionsAbfrage)
    // führt eine Aktionsabfrage aus.
    {
        $ok = mysql_query ($AktionsAbfrage, $this->Connection);
        if (!$ok)
            die ("Die Abfrage ist fehlgeschlagen: $AktionsAbfrage");
    }
    //*****************************************************
    function OpenRecordset ($AuswahlAbfrage)
    // führt eine Auswahlabfrage durch. Die Ergebnistabelle kann dann zeilenweise mit NextRecord gelesen werden. Ebenso kann die Anzahl der Tabellenzeilen mit RecordCount ermittelt werden
    {
        $this->Recordset = mysql_query ($AuswahlAbfrage, $this->Connection);
        if (!$this->Recordset)
            die ("Die Abfrage ist fehlgeschlagen: $AuswahlAbfrage");
    }
    ....
};
```

Entwicklung webbasierter Anwendungen

11. Kapitel: Technologien im Zusammenhang

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Prinzip


Webbrowser
Webserver
PHP-Programm
(bzw. CGI)
HTML Seite
1. Warenkorb
(xx.php) abfragen
HTML-Datei
mit
PHP-Anteil
2. PHP
starten
HTML Seite
(Ausgabe)
4. Ergebnis
übermitteln
HTML Seite
(evtl. mit
Script-Anteil)
3. erzeugte
HTML-Seite
DB