Grundlagen der Informatik 2

Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm

Hochschule Harz

FB Automatisierung und Informatik

mwilhelm@hs-harz.de

Raum 2.202

Tel. 03943 / 659 338

Gliederung

1. Einführung
2. WWW / HTML
   Überblick über WWW
   Einführung in HTML
3. Betriebssysteme (Aufgaben, Dateien, Speicher, Prozesse)
4. Unix
5. Unix Shellprogrammierung
6. PHP
7. **PHP und Datenbanken**

Dynamische Webseiten mit HTML, PHP, Mysql

- Ziel von PHP
  - Formularauswertung
  - Dateibearbeitung (Bilder)
  - Funktionen für das Upload
  - Anbindung an Datenbanken
- Aufbau der Sprache
- Was braucht man um mit PHP zu arbeiten?
  - Apache, PHP, Linux
  - Datenbank
  - WAMP
- Beispiele

Klassische Ansatz

Eigenschaften von PHP-Code

- Variablen
- Kontrollstrukturen (If, Case)
- Schleifen (While)
- Datentypen (Integer, Double, Bool, Strings)
- Arrays (einfache, multidim, indizierte, assozierte)
- Funktionen
- OOP, ab Version 5.0
- Anbindung an eine Datenbank
- Dateioperationen
- Dateiendungen (php, phtml)

Grundlagen

Ziele von Datenbank-Anwendungen

- Dynamisierung von Inhalten
- Möglichkeit der Abspeicherung von Eingabedaten aus dem Web
- Trennung von Daten, Anwendung und Darstellung
- Möglichkeit der Verwendung der gleichen Datenbasis für verschiedene Anwendungen

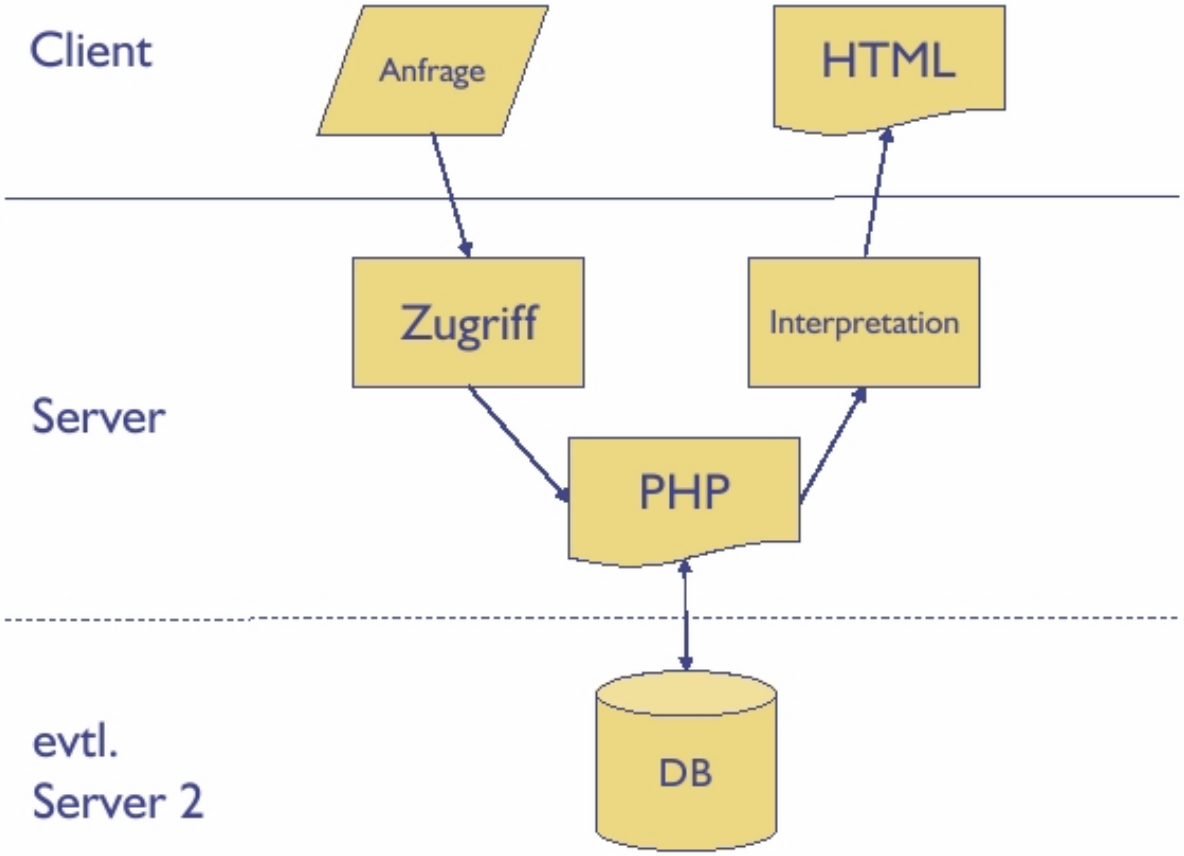
Grundlagen

Anbindung von Datenbanken an PHP

- PHP kann mit einer Vielzahl von Datenbankenkommunizieren:
  - MySQL
  - MS SQL
  - Oracle
  - PostgreSQL
  - mSQL
  - ODBC (-> DB2, MS Access u. a.)

Grundlagen

Ablauf einer Datenbank-Abfrage



Grundlagen

Voraussetzungen für die Beispiele

- Die folgenden Code-Beispiele verwenden den objektorientierten Syntax der verbesserten MySQL-Erweiterung (mysqli).
- Dieser erfordert:
  - PHP 5
  - MySQL 4.1
- Vorteile:
  - Funktionen werden Objektmethoden verwendet Statt (Objektorientierung)
  - Dadurch entsteht eine stärkere Unabhängigkeit von der verwendeten Datenbank
  - Höhere Geschwindigkeit

Erste Schritte

Initialisierung des Datenbank-Objekts

```
// Initialisierung des Objekts
$db = new mysqli(host, user,
   password, database);
// Prüfen, ob auch alles geklappt hat
if ( mysqli_connect_errno() )
{
   die('Initialisierung der Datenbank ist gescheitert');
}
```

Erste Schritte

Absenden eines SQL-Befehls

```
// Erst den SQL-Befehl in eine Variable speichern
// Übersichtlichkeit !
$sql = ?;
// Befehl senden
$result = $db->query($sql);
// Nun prüfen, ob der Befehl erfolgreich war
if ( !$result )
{
  // Es ist irgendetwas schief gegangen, abbrechen
  die('SQL-Anweisung gescheitert');
}
```

Datenbankabfragen

- Select-Befehl
  - Gibt eine „Matrix“ als Ergebnis zurück
  - Spaltenweise: Tupel
  - Pro Tupel die Attribute

- Insert-, Update-, Delete-Befehl
  - Gibt die Anzahl der betroffenen Tupel zurück

Datenbankabfragen

Rückgabewerte auslesen (1/2)

- Bei einigen SQL-Befehlen wird ein Ergebnis zurückgegeben (z. B. bei SELECT), andere geben jedoch kein Ergebnis zurück (z.B.INSERT)
- Dieses Ergebnis können wir mit `$row = $result->fetch_array()` abrufen.
- Dabei wird $row ein Array mit den Rückgabewerten zugewiesen. Die Rückgabewerte sind normalerweise über den Namen und über ihre Position zugänglich.

Datenbankabfragen

Rückgabewerte auslesen (2/2)

- Gibt z. B. ein SELECT mehrere Zeilen zurück, so erhalten wir mit dem Befehl jeweils nur eine Zeile. Rufen wir den Befehl erneut auf, so gibt er uns die nächste Zeile aus usw.
- Ist keine Zeile mehr vorhanden, so gibt fetch_array() FALSE zurück. Daher wird meist folgender Code verwendet:

```
while ( $row = $result->fetch_array() )
{
    // Dieser Teil wird für jede Zeile ausgegeben
}
```

Datenbankabfragen

Rückgabewerte auslesen - ein Beispiel

```
$sql = 'SELECT feld FROM tabelle';
if ( $result = $db->query($sql) )
{
  // Die Abfrage war erfolgreich!
  while ( $row = $result->fetch_array() )
  {
     // Ausgabe einer Zeile
     echo $row['feld'];
     echo $row[0];          // Gibt genau das Gleiche aus
  }
}
else
{
  // Die Abfrage war nicht erfolgreich
}
```

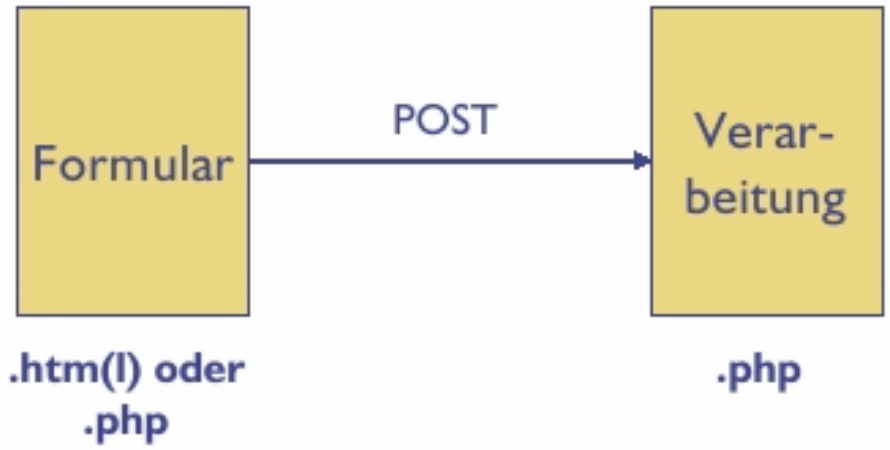
Übung

1. Aufgabe:
   - Erstellen Sie eine PHP Datei, die eine Übersicht über alle in einer Datenbank vorhandenen Bücher ausgibt.
   - Daten:
     - Host: localhost
     - Benutzer: u12345; Passwort: 67890
     - Datenbank: mydictionary
     - Tabelle: buecher
     - Felder: bueID, bueAutor, bueTitel, bueAuflage, bueJahr

Änderung von Daten

Einen neuen Datensatz erstellen

- Zum Erstellen eines neuen Datensatzes werden zwei Seiten verwendet:



Änderung von Daten

Das Formular

- Das Formular ist eine einfache, statische HTML-Seite. In ihm werden die Daten erfasst - also in unserem Beispiel:
  - Titel
  - Autor
  - Auflage
  - Jahr

Änderung von Daten

Die Verarbeitungsseite

- Die Verarbeitungsseite hat die Aufgabe, die vom Formular übermittelten Daten auszulesen ($_POST), daraus einen SQL-INSERT-Befehl zu erstellen und diesen an die Datenbank zu senden.
- Von der Verarbeitungsseite muss natürlich auch ein Link zurück vorgesehen werden.

Änderung von Daten

2. Aufgabe:

- Erstellen Sie ein Formular, mit dem neue Datensätze für die Datenbank erfasst werden können sowie die dazugehörige Verarbeitungsseite.
- Denken Sie dabei auch daran, dass das Formular von der Übersichtsseite aus Aufgabe 1 erreichbar sein muss und dass von der Verarbeitungsseite ein Link auf die Übersichtsseite führen muss.

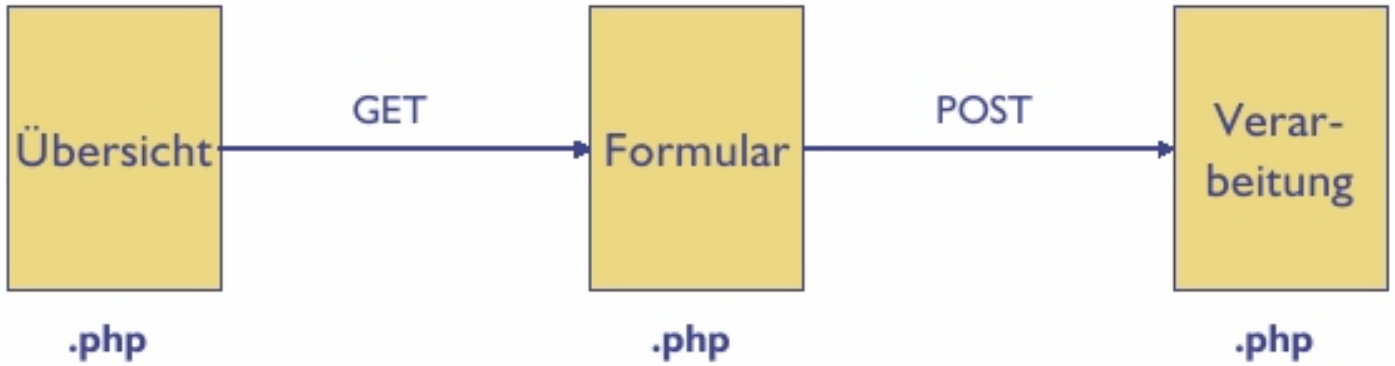
```php
<?php
        $db = new mysqli('localhost', 'web', 'web', 'web');
        if ( mysqli_connect_errno() ) {
                die('Initialisierung der DB gescheitert');
        }
        $bueAutor = ( isset($_POST['bueAutor']) ) ? $_POST['bueAutor'] : '';
        $bueTitel = ( isset($_POST['bueTitel']) ) ? $_POST['bueTitel'] : '';
        $bueAuflage = ( isset($_POST['bueAuflage']) &&
         $_POST['bueAuflage'] != '' ) ? intval($_POST['bueAuflage']) : 'NULL';
        $bueJahr = ( isset($_POST['bueJahr']) &&
        $_POST['bueJahr'] != ''  ) ? intval($_POST['bueJahr']) : 'NULL';

        $sql = "INSERT INTO buecher (bueAutor, bueTitel, bueAuflage, bueJahr)
                VALUES ('$bueAutor', '$bueTitel', $bueAuflage, $bueJahr)";
        if ( ! $db->query($sql) )       {
                die('Insert gescheitert');
        }
?>
```

Änderung von Daten

Einen Datensatz ändern

- Um einen Datensatz zu ändern, werden zwei Seiten verwendet, wobei die erste Seite von der Übersichtsseite aus die ID des zu ändernden Datensatzes erhalten muss:



Änderung von Daten

Das Formular

- Das Formular erhält von der Übersichtsseite die ID des Datensatzes. Sie ruft den entsprechenden Datensatz aus der Datenbank ab und erstellt ein Formular, dass die alten Daten enthält.
- Achtung: Die ID ist nicht änderbar und sollte daher auch nicht als Formularfeld ausgegeben werden. Allerdings muss sie an die Verarbeitungsseite weitergegeben werden, damit diese weiß, welcher Datensatz zu ändern ist.

Änderung von Daten

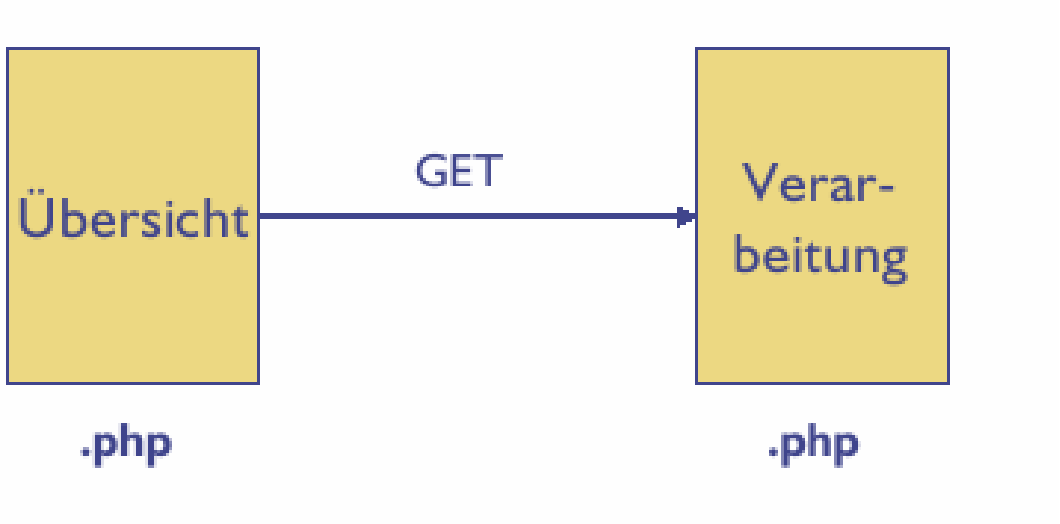
Die Verarbeitungsseite

- Die Verarbeitungsseite hat die Aufgabe, die vom Formular übermittelten Daten auszulesen ($_POST), daraus einen SQL-UPDATE-Befehl zu erstellen und diesen an die Datenbank zu senden.
- Von der Verarbeitungsseite muss natürlich auch ein Link zurück vorgesehen werden.

Änderung von Daten

Einen Datensatz löschen

- Wenn ein Datensatz gelöscht werden soll, ist dessen ID an eine Verarbeitungsseite zu übergeben, die dann den entsprechenden SQL DELETE Befehl erstellt.



Übung

3. Aufgabe:

- Erweitern Sie das Ergebnis von Aufgabe 1 und 2 so, dass auch eine Änderung und eine Löschung von Datensätzen möglich ist.

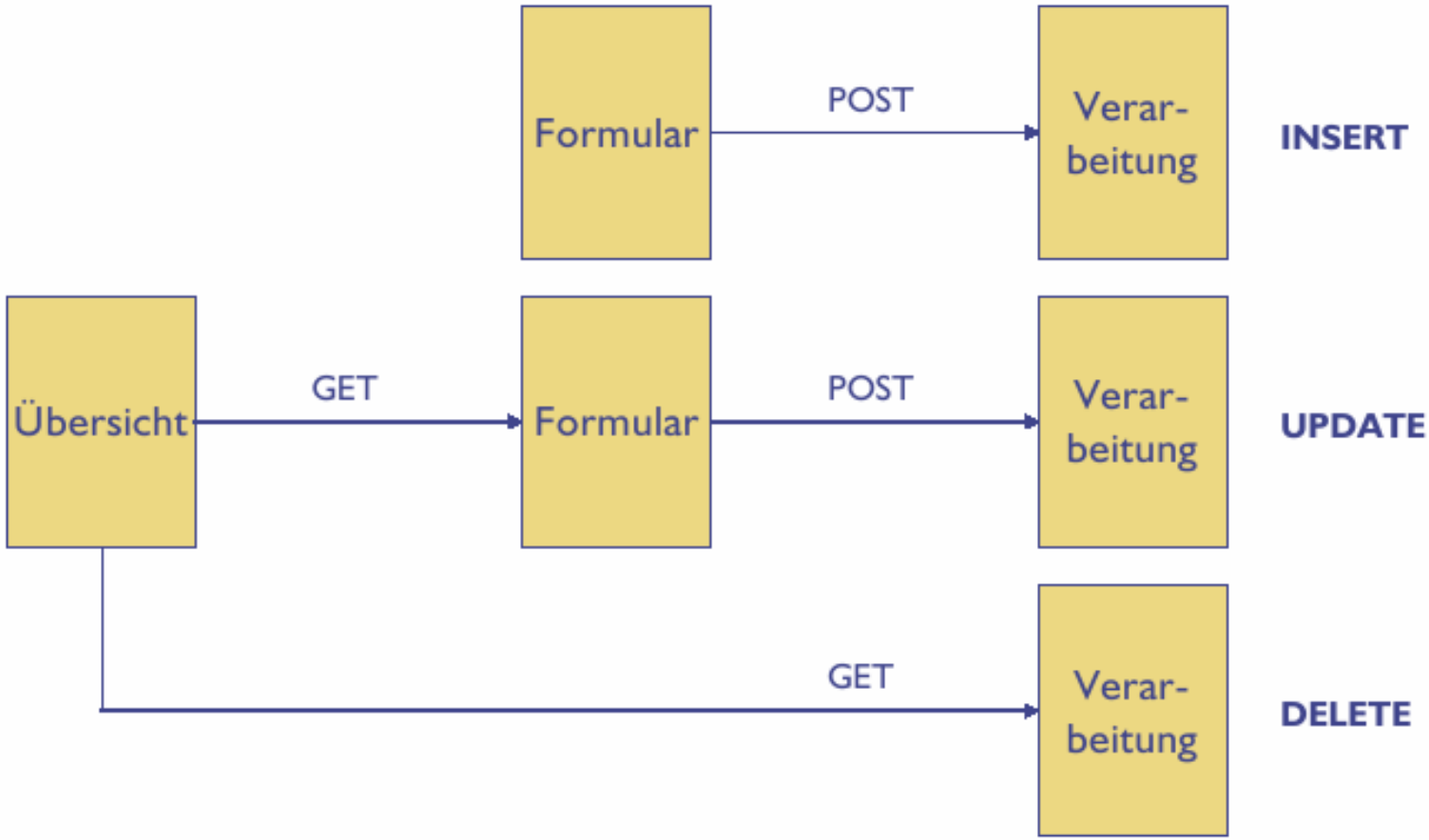
Änderung von Daten

Integration der Seiten

- In den meisten Fällen werden die Verarbeitungsroutinen in die Übersichtsseite integriert und oft auch die beiden Formulare zu einem zusammengefasst.
- Über geeignete Variablen müssen die Seiten dann ihr Verhalten an die jeweilige Situation anpassen.

Änderung von Daten

Integration der Seiten - Ausgangssituation



Änderung von Daten

Integration der Seiten