

Datenbanken & Webtechnologien

Prof. Dr. Andreas Hannig

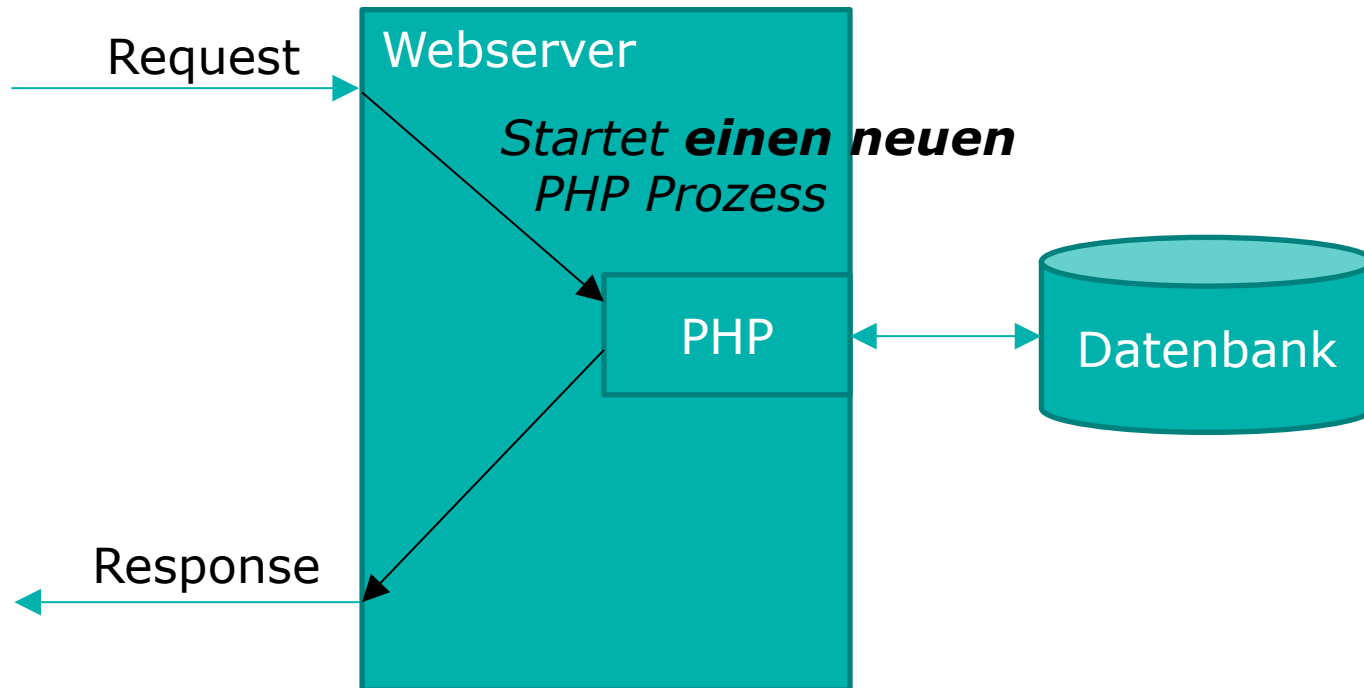
PHP Datenbankbindung mit mysqli

PHP

Datenbankanbindung mit mysqli

PHP

Datenbankanbindung

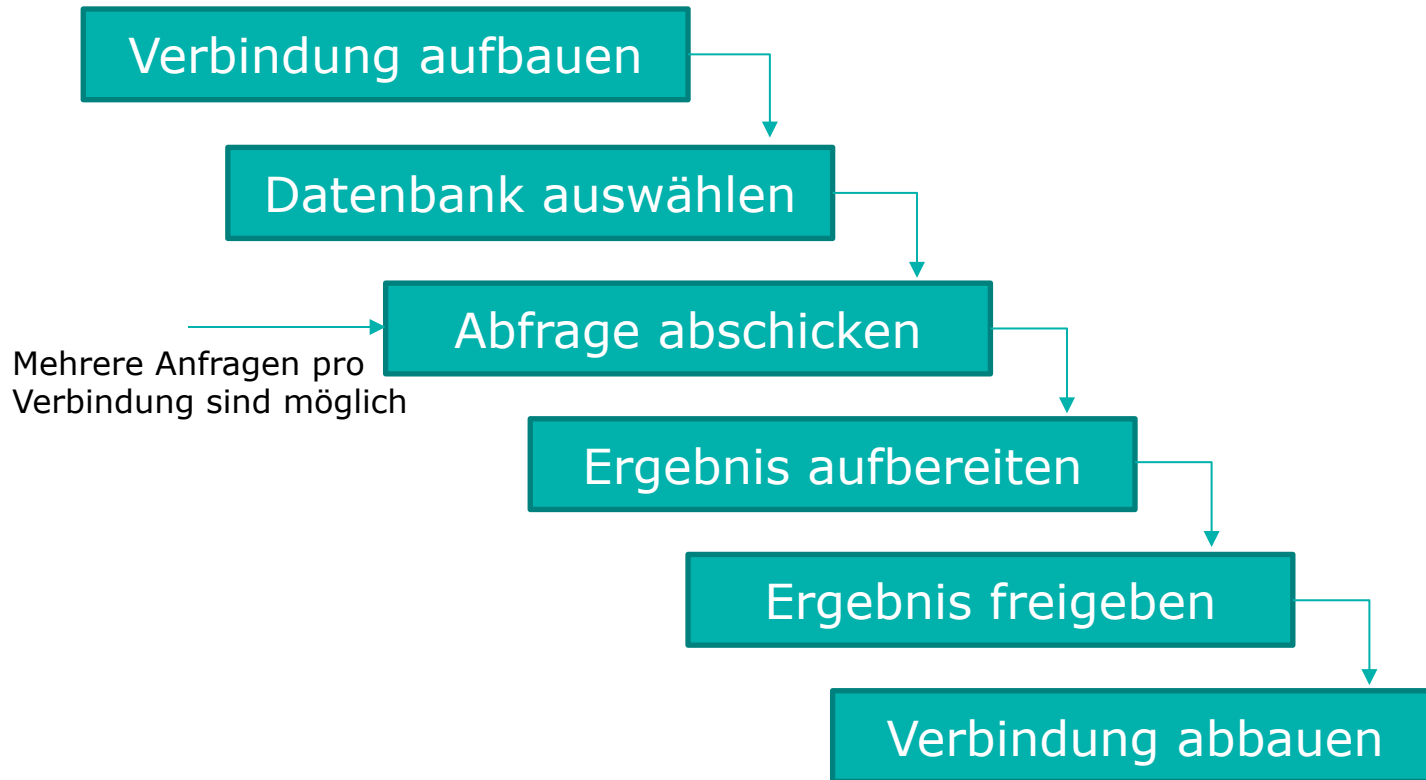


- Bei jedem Request müssen neue Verbindungen zur Datenbank aufgebaut werden, da der ausführende PHP Prozess nach der Verarbeitung **vollständig terminiert**.

PHP

Datenbankverbindung

- Eine Datenbankabfrage verläuft insgesamt in den folgenden **Schritten**:



PHP / Datenbankverbindung

Einführung

- Wir verwenden die PHP Funktionen `mysqli_*`
- Erinnerung: MariaDB basiert auf MySQL.
Die Kompatibilität (`mysql_ -> MariaDB`) ist gegeben.
- Über **Funktionen aufrufbar**.
Es sind keine Kenntnisse in der Objektorientierung notwendig.
- Für einfache Skripte ist der Funktionsumfang bereits ausreichend.
- Intern verwenden die `mysqli_*`-Funktionen die `mysqli`- Klasse und delegieren die Aufrufe lediglich weiter.
- Eine vollständige Dokumentation zu `mysqli_*` ist verfügbar unter:

MySQL Improved Extension

- [Einführung](#)
- [Overview](#)
- [Quick start guide](#)
 - [Dual procedural and object-oriented interface](#)
 - [Connections](#)
 - [Executing statements](#)
 - [Prepared Statements](#)
 - [Stored Procedures](#)
 - [Multiple Statements](#)
 - [API support for transactions](#)
 - [Metadata](#)
- [Installation/Konfiguration](#)
 - [Anforderungen](#)

<https://www.php.net/manual/de/book.mysqli.php>
(Letzter Zugriff 28.08.2021)

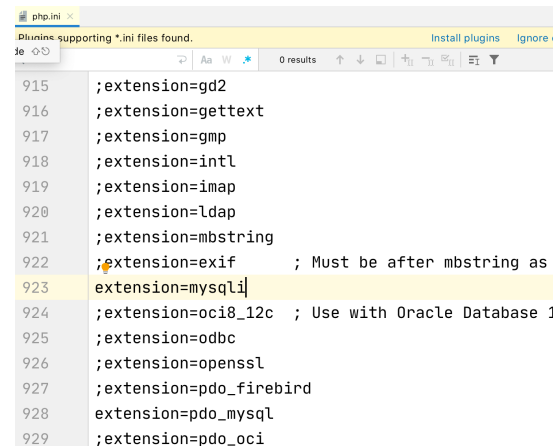
PHP / Datenbankverbindung

Einführung

- Zunächst muss die mysqli-Erweiterung in PHP aktiviert sein.
- Dafür müssen die folgenden Einträge in der php.ini freigeschaltet werden:

```
extension=mysqli  
extension=pdo_mysql
```

- Für ein Freischalten wird das vorangestellte „;“ (Semikolon) entfernt.



```
915 ;extension=gd2  
916 ;extension=gettext  
917 ;extension=gmp  
918 ;extension=intl  
919 ;extension=imap  
920 ;extension=ldap  
921 ;extension=mbstring  
922 ;extension=exif ; Must be after mbstring as  
923 extension=mysqli  
924 ;extension=oci8_12c ; Use with Oracle Database 1  
925 ;extension=odbc  
926 ;extension=openssl  
927 ;extension=pdo_firebird  
928 extension=pdo_mysql  
929 ;extension=pdo_oci
```

- Ansonsten kann es z.B. zu folgendem Fehler kommen:
„PHP Fatal error: Uncaught Error: Call to undefined function mysqli_connect()“
- Abhängig von Ihrer Installation kann es notwendig sein, die Extension mysqli selbst zu installieren

PHP / Datenbankverbindung

Verbindung aufbauen

```
$link = mysqli_connect(  
    "127.0.0.1",      // Host der Datenbank  
    "db-username",  // Benutzername zur Anmeldung  
    "db-password",  // Passwort zur Anmeldung  
    "my_db",        // Auswahl der Datenbank  
    $port);         // Optional: Port der Datenbank,  
                  // falls nicht 3306 verwendet wird
```

- `$link` ist eine Variable vom Typ „Ressource“. Es ist ein Handle für die Datenbank, der die aktive Verbindung beinhaltet.
- Ob eine Verbindung erfolgreich war, überprüft man mit:

```
if (!$link) {  
    echo "Verbindung fehlgeschlagen: ", mysqli_connect_error();  
    exit();  
}
```

- Tritt bei Verbindungsversuchen oder bei Abfragen **ein Fehler** auf, so geben die `mysqli_*`-Funktionen in der Regel `FALSE` zurück.
- Während der Verbindungsaufnahme stehen die Methoden zur Verfügung:

```
mysqli_connect_error()  
mysqli_connect_errno()
```

... die Funktionen geben z.B. eine Fehlermeldung zurück, wenn die Anmeldedaten (Benutzer und Passwort) nicht stimmen.

- Nach dem **erfolgreichen Verbindungsaufbau** und nach ersten Abfragen stehen Fehler über die folgenden Funktionen zur Verfügung:

```
mysqli_error($Link);  
mysqli_errno($Link);
```


PHP / Datenbankverbindung

Mögliche Fehlercodes von MariaDB

Code	Fehler	Beschreibung
1004	ER_CANT_CREATE_FILE	Can't create file '%s' (errno: %d)
1017	ER_FILE_NOT_FOUND	Can't find file: '%s' (errno: %d)
1101	ER_BLOB_CANT_HAVE_DEFAULT	BLOB/TEXT column '%s' can't have a default value
1111	ER_INVALID_GROUP_FUNC_USE	Invalid use of group function
1116	ER_TOO_MANY_TABLES	Too many tables; MariaDB can only use %d tables in a join
1146	ER_NO_SUCH_TABLE	Table '%s.%s' doesn't exist
...		

... insgesamt verwendet MariaDB nahezu 1000 verschiedene Fehlermeldungen. Die Fehlermeldungen geben einem Hinweise, wo Fehler innerhalb des verwendeten Statements vorhanden sind.

<https://mariadb.com/kb/en/library/mariadb-error-codes/> 01.10.2019

```
$sql = "SELECT name FROM rezept WHERE id = $id";
```

```
$result = mysqli_query($link, $sql);
```

- Gibt FALSE zurück bei einem Fehler. Bei erfolgreichem SELECT, SHOW oder EXPLAIN ist `$result` ein Objekt vom Typ `mysqli_result`. Ansonsten ist `$result` TRUE.

PHP / Datenbankverbindung

Ergebnisse aufbereiten

rezept

id	name	beschreibung
1	Spaghetti Bolognese	Nudelgericht mit Hackfleisch
2	Pilzresotto	Champignons, Zwiebeln und Knoblauch

```
$sql = "SELECT name, beschreibung FROM rezept ORDER BY id";  
$result = mysqli_query($link, $sql);
```

```
while ($row = mysqli_fetch_row($result)) {  
    var_dump($row);  
}
```

```
array(2) {  
    [0]=>string(19) "Spaghetti Bolognese"  
    [1]=>string(28) "Nudelgericht mit Hackfleisch"  
}  
array(2) {  
    [0]=>string(11) "Pilzresotto"  
    [1]=>string(35) "Champignons, Zwiebeln und Knoblauch"  
}
```

<https://www.php.net/manual/en/mysqli.query.php> 20.09.2020

PHP / Datenbankverbindung

Ergebnisse aufbereiten

```
$sql = "SELECT name, beschreibung FROM rezept ORDER BY id";
```

- Einzelne Zeilen eines Ergebnisses können mit unterschiedlichen Methoden gelesen werden:

Methoden	Beschreibung
<code>mysqli_fetch_row</code>	Arrayschlüssel sind die Feldnummern. <pre>array(1) { [0]=>string(19) "Spaghetti Bolognese" }</pre>
<code>mysqli_fetch_assoc</code>	Arrayschlüssel sind die Feldnamen. <pre>array(1) { ["name"]=>string(19) "Spaghetti Bolognese" }</pre>
<code>mysqli_fetch_array</code>	Arrayschlüssel sind Feldnummern und -namen. <pre>array(1) { [0]=>string(19) "Spaghetti Bolognese", ["name"]=>string(19) "Spaghetti Bolognese" }</pre>

PHP / Datenbankverbindung

Beispiel: Ergebnisse aufbereiten

- Das Beispiel zeigt, wie Ergebnisse abgefragt und als HTML-Liste () dargestellt werden können.

```
$sql = "SELECT id, name, beschreibung FROM rezept";
$result = mysqli_query($link, $sql);

echo '<ul>';

while ($row = mysqli_fetch_assoc($result)) {
    echo '<li>',
        '<a href="rezept.php?id=' . $row['id'] . ">',
        $row['name'],
        '</a>',
        ' - ',
        $row['beschreibung'], '</li>';
}

echo '</ul>';
```

```
$sql = 'UPDATE rezept SET name="Pilzresotto 2" WHERE id=2';
```

```
mysqli_query($link, $sql);
```






```
$num = mysqli_affected_rows($link);
```

- `mysqli_query` gibt bei erfolgreicher Ausführung TRUE zurück
- Nach dem Hinzufügen, Manipulieren oder Löschen von Daten (INSERT, UPDATE, DELETE, ...) gibt die Methode `mysqli_affected_rows` die Anzahl der betroffenen Datensätze zurück. Bei einem Fehler: -1.

PHP / Datenbankverbindung

Letzte verwendete ID erhalten

- Erzeugen wir ein Attribut mit AUTO INCREMENT, so wird beim Einfügen durch die Datenbank eine ID automatisch vergeben.
- Dies kommt oft zur automatischen Vergabe bei Werten von Primärschlüsseln zum Einsatz.
- Die Funktion `mysqli_insert_id` ermöglicht eine Abfrage der ID, die automatisch von der Datenbank vergeben wurde.

```
▼  gericht
  ▼  columns 9
     id int(10) unsigned (auto increment)
     name varchar(80)
     beschreibung varchar(800)
```

```
$sql = "INSERT INTO gericht (name, beschreibung) " .
      " VALUES ('Test 42', 'f 42');";
if(mysqli_query($link, $sql)) { // Prüfe, ob das
                               // Einfügen erfolgreich war.
    $id = mysqli_insert_id($link);
    echo "Von der Datenbank wurde die Id: ".
        "$id automatisch vergeben";
}
```

<https://www.php.net/manual/de/mysqli.insert-id.php> 06.09.2021

PHP / Datenbankverbindung

Datenabfrage mit prepared Statements

```
$sql = "SELECT name FROM rezept WHERE id = $id";
```

- Der einfache Zusammenbau von Statements mit einer String-Konkatenation kann zu Sicherheitsproblemen führen: SQL-Injections (→ Später: Thema Security)
- **Sicherer** und bei mehrfacher Ausführung **effizienter** sind „prepared Statements“.
- Dies gilt auch für UPDATE-, DELETE- und INSERT-Statements.

<https://www.php.net/manual/en/mysqli.query.php> 20.09.2021

PHP / Datenbankverbindung

Beispiel: Datenabfrage mit prepared Statements

```
$sql = "SELECT name FROM rezept WHERE id = ?";  
$bind_param = 'i';
```

```
$statement = mysqli_stmt_init($link);  
mysqli_stmt_prepare($statement, $sql);
```

```
mysqli_stmt_bind_param($statement, $bind_param, $id);
```

```
mysqli_stmt_execute($statement);  
$result = mysqli_stmt_get_result($statement);
```

```
$data = mysqli_fetch_array($result);
```

- bind_param kennt die Typen:
 - i Integer
 - d Double
 - s String
 - b Blob

<https://www.php.net/manual/en/mysqli.prepare.php> 20.09.2021

PHP / Datenbankverbindung

Ergebnis freigeben

- Zur Schonung der Ressourcen ist der verwendete Speicher nach einer erfolgreichen Verarbeitung wieder freizugeben.

```
$result = mysqli_query($link, /* SQL */);  
if($result) {  
    // Verarbeitung  
    mysqli_free_result($result);  
}
```

- *In PHP ist das nicht ganz so wichtig, da der Prozess nach der Bearbeitung terminiert (und der Speicher damit freigegeben wird). Jedoch bei **größeren Massenverarbeitungen** ist dies von großer Bedeutung.*

PHP / Datenbankverbindung

Verbindung abbauen

- Nach der Kommunikation mit der Datenbank ist die Verbindung zu schließen.

```
mysqli_close($link);
```

PHP / Datenbankverbindung

Vollständiges Beispiel: Abfrage

```
<?php
$link=mysqli_connect("localhost", // Host der Datenbank
    "root", // Benutzername zur Anmeldung
    "Datenbankpasswort", // Passwort
    "emensawerbeseite" // Auswahl der Datenbanken (bzw. des Schemas)
    // optional Port der Datenbank
);

if (!$link) {
    echo "Verbindung fehlgeschlagen: ", mysqli_connect_error();
    exit();
}

$sql = "SELECT id, name, beschreibung FROM gericht";

$result = mysqli_query($link, $sql);
if (!$result) {
    echo "Fehler während der Abfrage: ", mysqli_error($link);
    exit();
}

while ($row = mysqli_fetch_assoc($result)) {
    echo '<li>', $row['id'], ':', $row['name'], '</li>';
}

mysqli_free_result($result);
mysqli_close($link);
```

Datenbanken und Webtechnologien

FH Aachen

Fachbereich Elektrotechnik und Informationstechnik

Prof. Dr. Andreas Hannig

Lehrbereich Datenbanken und Business Intelligence