Hintergrund

Basis-Komponenten für Webauftritte

- Linux/Windows *Betriebssystem*
- **A**pache *Webserver*
- **M**ySQL/**M**ariaDB *Datenbank*
- **P**HP/**P**erl
  Laufzeit-Umgebung für eine Programmiersprache

Wenn Sie Webspace mieten, erhalten Sie meist ein LAMP-System.

Apache

- Webserver liefern Webseiten aus.
- Apache ist der verbreitetste Webserver.
+ Apache erfüllt auch komplexe Anforderungen.
+ Apache ist umfangreich konfigurierbar.
– Die Konfigurationsregeln sind kompliziert.
– Es gibt schnellere Webserver.

MySQL und MariaDB

- MySQL ist eine schnelle, mehrbenutzerfähige Datenbank.
- MySQL hat einige Abweichungen vom SQL-Standard.
- MySQL war dual lizenzierbar.
- Nachdem MySQL von Oracle gekauft wurde, entstand der freie Fork MariaDB.

CGI und Laufzeit-Module

- CGI (Common Gateway Interface) ist die klassische Schnittstelle von Webservern zu Programmen, die Seiten erzeugen.
- CGI kann mit Programmen in beliebigen Sprachen zusammenarbeiten.
- CGI ist langsam, weil bei jedem Seitenaufruf ein Programm gestartet werden muss.
- Sprachen wie PHP werden durch Apache-Module ausgeführt und sind dadurch schneller.

PHP

- PHP ist eine domänenspezifische Sprache zur serverseitigen Webprogrammierung.
- PHP entstand aus einem Hobbyprojekt und entwickelte sich allmählich zu einer umfangreichen objektorientierten Sprache.
- Zur Zeit ist PHP 7 noch sehr verbreitet.

XAMPP

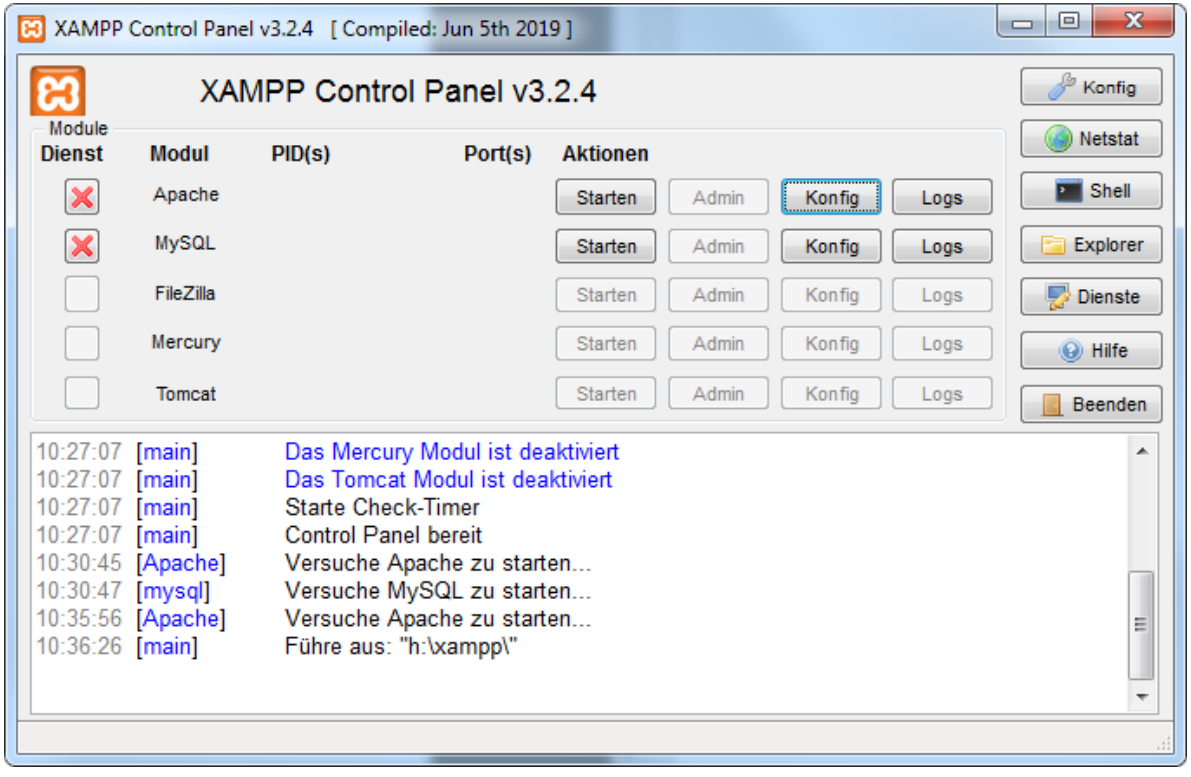
- XAMPP ist ein Software-Stack für die Webentwicklung.
- Alle benötigten Komponenten sind komfortabel vorkonfiguriert.
- Für den Produktivbetrieb ist XAMPP nicht geeignet (zu unsicher).

XAMPP installieren

- XAMPP installiert sich in einem Ordner Ihrer Wahl (typisch: `C:\XAMPP`), nicht im Windows-Programmverzeichnis.
- In diesem Ordner liegen Programmdateien und Daten.
- Für den Unterricht benötigen wir nur Apache und MySQL. FileZilla, Mercury und Tomcat sind entbehrlich.

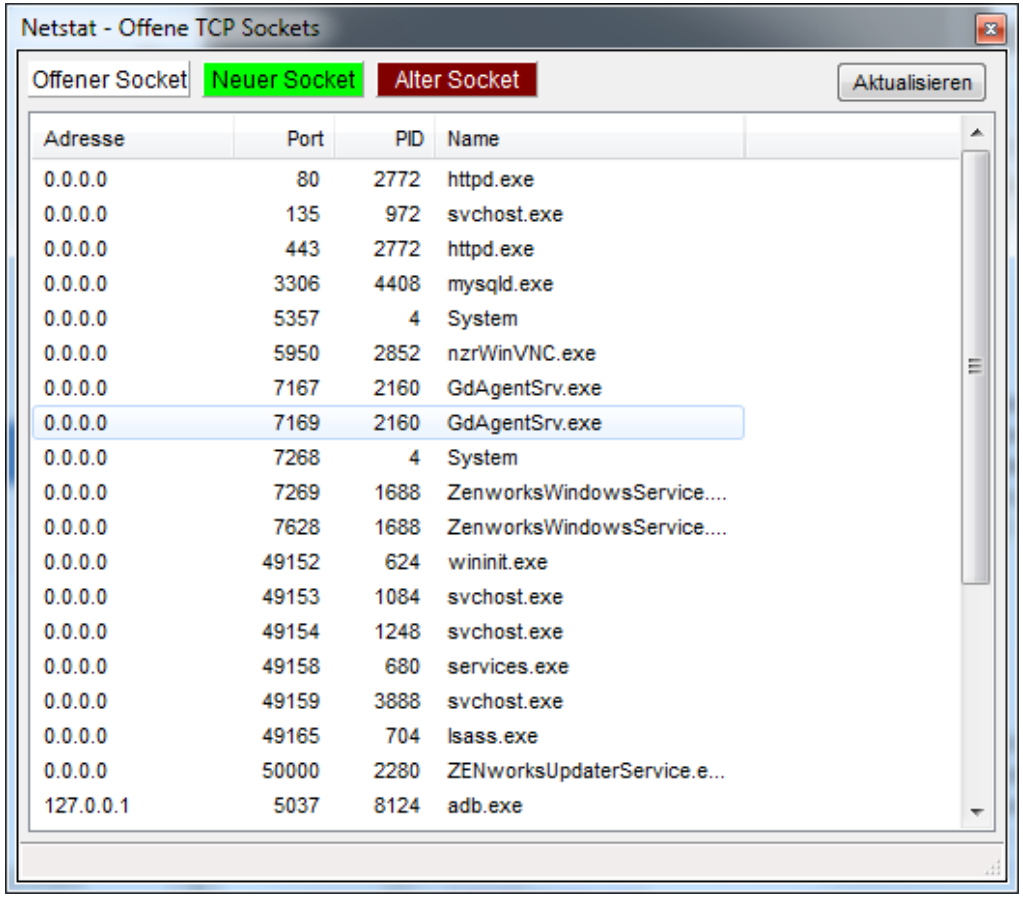
XAMPP bedienen

- Apache und MariaDB werden durch das XAMPP-Controlpanel oder eine Batchdatei gestartet.
- Sie werden durch eine Batchdatei beendet.
- Sie können als Windows-Dienste konfiguriert werden. Das ist nur sinnvoll, wenn Sie sie immer benötigen.



XAMPP bedienen

- Im XAMPP-Controlpanel können Sie mit Netstat prüfen, ob Apache und MariaDB laufen.



MariaDB und MySQL: Syntax-Besonderheiten

- Bei Tabellennamen wird Groß- und Kleinschreibung in Abhängigkeit vom Betriebssystem beachtet.
  Schreiben Sie Tabellennamen immer klein.
- In MariaDB müssen Sie jeden SQL-Befehl mit einem `;` beenden.
- Befehle zur Steuerung der Sitzung beginnen mit einem `\`.
- Es gibt weitere Abweichungen zur üblichen SQL-Syntax, z.B. bei `||`.

MariaDB benutzen

- Kommandozeilenwerkzeuge liegen im Ordner `XAMPP\mysql\bin`.
- Den grafischen Client phpMyAdmin erreichen Sie mit einem Webbrowser unter `http://localhost/phpmyadmin` oder `http://127.0.0.1/phpmyadmin`
- Wir müssen teilweise die Kommandozeile benutzen, weil es in XAMPP nicht einfach möglich ist, den phpMyAdmin-Benutzer zu wechseln.

MariaDB Kommandozeilen-Client starten und beenden

- Zuerst starten Sie eine Eingabeaufforderung:
  `cmd` ins Startmenü tippen.
- `c:\XAMPP\mysql\bin\mysql -u` *Benutzername* `-p`.
- Beim ersten Start verwenden Sie `root` als Benutzername und lassen das `-p` weg.
- Den Client beenden Sie mit `\q`.

MariaDB Benutzer und Datenbank anlegen

Laden Sie die Datei abrechnung.sql aus dem Tauschordner in das aktuelle Verzeichnis.
Mit dem folgenden Befehl legen Sie daraus eine Datenbank an und füllen sie mit Daten:

```
c:\XAMPP\mysql\bin\mysql -u root < abrechnung.sql
```

SQL-Batchdateien

- `Abrechnung.sql` ist eine SQL-Batchdatei; Sie können sie z.B. mit Notepadd++ ansehen.
- Batchdateien enthalten mehrere Befehle, die nacheinander ausgeführt werden.
- In SQL-Batchdateien werden zwei Minuszeichen als Kommentarzeichen verwendet:
  ab -- wird alles bis zum Zeilenende ignoriert.
- Wenn Sie Batchdateien erstellen, nutzen Sie Kommentare zur Dokumentation.
- Damit die Batchdatei im Fehlerfall nicht teilweise ausgeführt wird, packt man den Inhalt in eine Transaktion `(Start Transaction … commit)`

Schema

- Tabellen liegen in einem Namespace oder *Schema*.
- Bei manchen Anbietern werden Schema und Datenbank gleich behandelt.
- Objekte in einem anderen Schema werden angesprochen, indem der Schemaname und ein Punkt vor den Objektnamen gestellt werden.

```
SELECT * FROM Produktion.Mitarbeiter
```

Benutzer

- Anmeldung an der Datenbank mit Benutzername und Passwort.
- Benutzer anlegen mit `CREATE USER '`*Name*`' IDENTIFIED BY '`*Passwort*`'`
- Benutzer löschen mit DROP USER
- Rechtevergabe auf Schema- oder Tabellenebene:
  `GRANT SELECT ON Gehalt TO Personaler`
- Rechteentzug mit REVOKE:
  `REVOKE ALL ON Gehalt FROM Fred`

Trigger

- Trigger sind Aktionen, die vor oder nach dem Anlegen, Ändern und Löschen von Datensätzen automatisch ablaufen.
- Sie werden zum Beispiel zur Überwachung von Tabellen verwendet.
- Trigger werden in SQL geschrieben.

Trigger

- Trigger anlegen mit CREATE TRIGGER.
- Trigger löschen mit DROP TRIGGER.
- Es gibt BEFORE- und AFTER-Trigger.
- Es gibt je Ereignis (INSERT, UPDATE, DELETE) maximal einen Before- und einen After-Trigger.
- Die Schlüsselwörter old/new referieren den alten/neuen Datenstand.

Beispiel: Dokumentieren von Gehaltsänderungen

Problembeschreibung

Es gab Unregelmäßigkeiten bei der Abrechnung von Gehältern. Zur Klärung der Ursache sollen Änderungen in der Gehaltstabelle überwacht werden.

Beispiel: Dokumentieren von Gehaltsänderungen

```
DELIMITER //
CREATE TRIGGER Gehaltupd
AFTER UPDATE ON Abrechnung.Gehalt
FOR EACH ROW
BEGIN
  insert into abrechnung.gehaltsaenderung
  (zeitpunkt, benutzer, mitarbeiternummer,
    monat, gehaltalt, gehaltneu)
  values (now(), current_user(),
new.mitarbeiternummer,
    new.monat, old.betrag, new.betrag);
END //
DELIMITER ;
```

Erklärung zum Beispiel

- Das Befehlsende muss vor der Triggerdefinition geändert werden, damit der Parser nicht denkt, dass der Befehl beim ersten Stichpunkt endet.
  `DELIMITER //`
- Nach der Definition muss das Befehlsende wieder zurückgesetzt werden.
  `DELIMITER ;`
- Datum und Uhrzeit der Änderung werden mit `NOW()` abgefragt.
- Der Datenbankbenutzer wird mit `CURRENT_USER()` ermittelt.
- Das alte Gehalt kann mit `OLD.Betrag` erfragt werden, das neue mit `NEW.Betrag`.

Stored Procedure

- Stored Procedures sind (kleine) Programme, die von der Datenbank ausgeführt werden.
- Anlegen mit `CREATE PROCEDURE` *Name*, löschen mit `DROP Procedure` *Name*, Aufrufen mit `CALL` *Name*`()`
- Die Möglichkeiten, Programmiersprachen usw. unterscheiden sich je nach Datenbank-Engine.
- Beispiele
  - Zusammenfassen mehrerer Befehle.
  - Versenden einer E-Mail.

Beispiel: Mitarbeiter mit Gehalt anlegen

Problembeschreibung

Neue Mitarbeiter werden in mehreren Schritten angelegt:

- Erzeugen einer eindeutigen Mitarbeiternummer.
- Eintragen in der Mitarbeiter-Tabelle.
- Eintragen in der Gehalts-Tabelle.

Kein Schritt darf vergessen werden.

Beispiel: Mitarbeiter mit Gehalt anlegen

```
DELIMITER //
CREATE PROCEDURE LegeMitarbeiterAn(
  IN nachname VARCHAR(30),
  IN vorname VARCHAR(30),
  IN anfangsgehalt INTEGER);
BEGIN
  DECLARE manummer INT DEFAULT 0;
  SELECT MAX(mitarbeiternummer)
    INTO manummer FROM mitarbeiter;
  SET manummer = manummer + 1;
  INSERT INTO mitarbeiter VALUES(manummer,
    nachname, vorname);
  INSERT INTO gehalt VALUES(manummer,
    YEAR(NOW())*100 + MONTH(NOW()), anfangsgehalt);
END //
DELIMITER ;
```

Erklärung zum Beispiel

- Das Befehlsende muss vor der Triggerdefinition geändert und nachher zurückgesetzt werden (DELIMITER).
- Der Prozedur werden drei Variablen übergeben: Nachname, Vorname und Anfangsgehalt.
- Eine Variable *manummer* wird definiert.
- Sie erhält die höchste Mitarbeiternummer. Diese wird um 1 erhöht und dient dann als Nummer des neuen Mitarbeiters.
- Angaben zum neuen Mitarbeiter werden in die Mitarbeiter- und in die Gehaltstabelle eingetragen.
- Der Monat wird aus dem aktuellen Datum berechnet. (Das ergibt 202109 für ein Datum im September 2021.)