PHP unter Ubuntu Linux mit den Eclipse PDTs

1. Zuallererst müssen der Apache2 Web Server und die PHP-Unterstützung installiert werden:
   ```
   sudo apt install apache2 php libapache2-mod-php php-mbstring php-mysql
   ```
2. Da Eclipse PDT die Projekte unter einer Basis-URL wie http://localhost:8080 erwartet, müssen wir eine neue Site (= Virtual Host) anlegen.
   Im einfachsten Fall wird das Projektverzeichnis gleich dem Eclipse Workspace sein, d.h. */home/user/workspace*
   ```
   sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/php-workspace.conf
   ```
3. Die Site anpassen:
   ```
   sudoedit /etc/apache2/sites-available/php-workspace.conf
   ```
   dabei
   `VirtualHost` auf `*:8080`
   `DocumentRoot` auf `/home/user/workspace`
   und das Projektverzeichnis für den Webserver freigeben (am Ende der Datei hinzufügen):
   ```
   <Directory /home/user/workspace/>
               Options Indexes FollowSymLinks
               AllowOverride None
               Require all granted
   </Directory>
   ```
4. Die Site aktivieren:
   ```
   sudo a2ensite php-workspace
   ```
5. Es muss der neue Endpoint 8080 erfasst werden:
   ```
   sudoedit /etc/apache2/ports.conf
   ```
   Dazu muss unter Listen 80 einfach
   ```
   Listen 127.0.0.1:8080
   ```
   hinzugefügt werden.
   Empfehlenswert: Vor alle Listen-Direktiven das Präfix `127.0.0.1:` setzen, um zu verhindern, dass der Server von außen zugreifbar bleibt.
6. Dem Homeverzeichnis Navigationsberechtigungen für den Webserver erteilen:
   ```
   chmod o+x ~
   ```
7. Diagnosedatei anlegen:
   ```
   echo '<?php phpinfo(); ?>' > /home/user/workspace/phpinfo.php
   ```
8. Server neu starten, danach sollte die URL http://localhost:8080/phpinfo.php eine Ausgabe liefern:
   ```
   sudo systemctl restart apache2.service
   ```

PHP Version 7.2.24-0ubuntu0.18.04.1

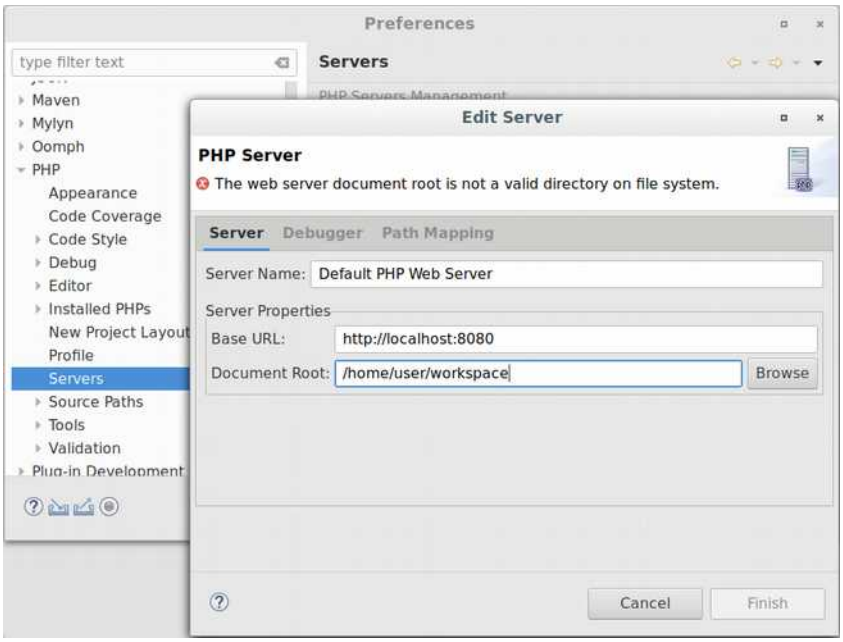
System	Linux mdw-InfinityBook13V3 4.15.0-74-generic #84-Ubuntu SMP Thu Dec 19 08:06:28 UTC 2019 x86_64
Build Date	Oct 28 2019 12:07:07
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini

9. Jetzt bleibt nichts anderes übrig, als Eclipse PDT anzupassen. D.h. in Eclipse den Menüpunkt *Window/Preferences* öffnen und unter dem Punkt *PHP* den Unterpunkt *Servers* auswählen. Den Server auswählen und mittels Edit die Parameter korrekt einstellen:

   `Base URL` auf http://localhost:8080

   `Document Root` auf */home/user/workspace*

10. Nun einfach ein PHP-Skript wie *index.php* in einem Projekt auswählen und mittels *Run As →/PHP*



Web Application zur Ausführung bringen.

11. Mittels `sudo tail -f /var/log/apache2/error.log` sind die bei der Ausführung entstandenen (PHP-)Fehlermeldungen einsehbar.

Quellen

- Ubuntu-Dokumentation: https://ubuntu.com/server/docs/web-servers-apache und https://ubuntu.com/server/docs/programming-php
- Eclipse PDT-Dokumentation: https://help.eclipse.org/topic/org.eclipse.php.help/html/table_of_contents.html

MariaDB/MySQL und phpMyAdmin

MySQL vs. MariaDB

MySQL wird schon seit vielen Jahren vom US-Konzern *Oracle* weiterentwickelt, welcher es zwar weiterhin in mehr oder weniger regelmäßigen Abständen in einer offenen Variante zur Verfügung stellt, aber primär auch am Vertrieb der proprietären Version Interesse zeigt. Vor allem die letzte Version 8.0 zeichnet sich durch einen bedeutenden Qualitätssprung aus, da sie über das „Data Dictionary“ Crash-sichere Schema-Manipulationen erlaubt[1].

MariaDB hingegen sieht sich als legitimer Nachkomme der ursprünglichen MySQL Company und wird weiterhin von Begründer Michael Widenius im Rahmen der Stiftung MariaDB Foundation betreut. Für diese Organisation steht absolute Transparenz bei neuen Features und Fehlern im Vordergrund, und neue Versionen erscheinen in regelmäßigen Abständen. Die finnische Firma MariaDB Corporation AB bietet Geschäftskunden Support. Derzeit (2021) hinkt MariaDB MySQL in punkto Crash-sicherem Schema noch hinterher, dessen Umsetzung befindet sich aber in Ausarbeitung[2].

Was GNU/Linux betrifft bieten die meisten Distributionen in ihren Repositorien nur mehr MariaDB an, andere wie Ubuntu stellen beide Variationen zur Verfügung[3]. Unter Windows und MacOS X stellt sich diese Frage hingegen nicht, da man dort so und anders den Server aus Drittquellen bezieht.

Zur Installation folgenden Befehl nutzen:

```
sudo apt install mariadb-server
```
bzw.
```
sudo apt install mysql-server
```

Anschließend einen Admin-Benutzer anlegen, welcher nicht `root` heißen soll (Beispiel: Benutzername `admin`, Kennwort `masterkey`) [4]:

```
sudo mysql
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' IDENTIFIED BY 'masterkey' WITH GRANT OPTION;
```

Nun einfach die MariaDB/MySQL-Konsole mittels *Strg+D* beenden.

phpMyAdmin

Grundsätzlich gibt es unzählige Tools zum Verwalten von SQL-basierten Datenbanksystemen. Die Bandbreite reicht von großen kommerziellen Tools (*Aqua Data Studio*, www.aqualfold.com) bis hin zu freien Lösungen (*SquirreL SQL*, http://www.squirrelsql.org/ oder *Heidi*, https://www.heidisql.com/), die mit allen gängigen DBMS zusammenarbeiten. Speziell für MariaDB/MySQL gibt es die *MySQL Workbench* (https://mysqlworkbench.org/, auch in den Paketrepositorien) und das webbasierte *phpMyAdmin*, das als unsere Referenzplattform dient[5].

1. phpMyAdmin hat den großen Vorteil, dass es als Paket vorkonfiguriert in allen gängigen Distributionen zur Verfügung steht und kann daher über folgenden Befehl installiert werden:
   ```
   sudo apt install phpmyadmin
   ```

1 https://dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html
2 https://mariadb.org/documentation/ und https://mariadb.com/legal/
3 https://de.wikipedia.org/wiki/MariaDB#Einzelnachweise, https://de.wikipedia.org/wiki/MySQL#Kritik
4 https://mariadb.com/kb/en/configuring-mariadb-for-remote-client-access/#granting-user-connections-from-remote-hosts, root ist unter GNU/Linux lediglich für die Konsolenanmeldung vorgesehen
5 https://www.phpmyadmin.net/

2. Im Paketkonfigurationsdialog sich für `apache2` als Webserver entscheiden und bei der DB-Konfiguration für `dbconfig-common` optieren.
3. Das phpMyAdmin-Kennwort leer lassen, damit es automatisch vergeben wird, ebenso das Passwort für den Benutzer *root*.
4. Falls der Dialog zur Auswahl der Verbindungsmethode angezeigt wird, sich für `Unix-Socket` entscheiden.
5. Wenn alles gut gegangen ist, steht unter http://localhost/phpmyadmin/ eine funktionsfähige phpMyAdmin-Instanz bereit.
   Die Anmeldung mit dem zuvor angelegten Administratorkonto sollte klappen, so dass folgende Benutzeroberfläche angezeigt werden sollte: