macOS

MAMP (Aktivierung + Komplettierung)

Da XAMPP scheinbar unter reinen 64-Bit-Systemen (OS X und Linux) seit einiger Zeit ein paar Probleme hat, habe ich bereits zu Zeiten von OS X 10.11/El Capitan (akutell macOS 10.15/Catalina) nach einer anderen Lösung suchen müssen.

Da sowohl Apache (aktuell 2.4.41) als auch PHP (aktuell 7.3.11) bei OS X seit geraumer Zeit mit an Bord sind, möchte ich sie nutzen. Hierzu brauchen sie nur „etwas“ umkonfiguriert zu werden.

Zusätzliche Apache HTTPD-Module (PHP und userdir) aktivieren

```
$ sudo sed -E -e 's!(^#)(LoadModule\ userdir_module\
libexec/apache2/mod_userdir.so)!\2!' -i "" /private/etc/apache2/httpd.conf
$ sudo sed -E -e 's!(^#)(LoadModule\ php7_module\
libexec/apache2/libphp7.so)!\2!' -i "" /private/etc/apache2/httpd.conf
$ sudo sed -E -e 's!(^#)(Include\ /private/etc/apache2/extra/httpd-
userdir.conf)!\2!' -i "" /private/etc/apache2/httpd.conf
```

Im Folgenden gehe ich davon aus, dass Dein Benutzer-Login „user“ (also ggf. anpassen!) ist!

"Benutzerverzeichnisse" konfigurieren

```
$ sudo sh -c 'cat <<EOF >> /private/etc/apache2/extra/httpd-userdir.conf

# added by user:
<Directory "/Users/user/Sites/">
  AllowOverride All
  Options Indexes FollowSymLinks MultiViews
  Require all granted
</Directory>
EOF'
```

PHP für die per Homebrew nachinstallierte MariaDB vorbereiten

```
$ sudo cp /private/etc/php.ini.default /private/etc/php.ini
$ sudo sed -e 's!pdo_mysql.default_socket=!&\ /tmp/mysql.sock!' -i ""
/private/etc/php.ini
$ sudo sed -e 's!mysql.default_socket\ =!&\ /tmp/mysql.sock!' -i ""
/private/etc/php.ini
$ sudo sed -e 's!mysqli.default_socket\ =!&\ /tmp/mysql.sock!' -i ""
/private/etc/php.ini
```

Apache HTTP-Server starten

Da der Apache bei mir standardmäßig auf dem Client nicht automatisch gestartet wird, starte ich ihn jetzt:

```
$ sudo apachectl start
```

Sollte er bei Dir aus irgend einem Grund bereits laufen, einfach neustarten:

```
$ sudo apachectl restart
```

1. Test

Nun noch schnell eine einfache Datei für Testzwecke anlegen:

```
$ mkdir ~/Sites
$ echo "<?php phpinfo(); ?>" > ~/Sites/phpinfo.php
```

Und Alles bis hier (also noch ohne MariaDB) testen:

```
$ open -a Safari "http://localhost/~user/phpinfo.php"
```

Nach ein paar Jahren MacPorts (https://www.macports.org/) bin ich nun bei Homebrew gelandet.

Homebrew installieren: http://brew.sh/index_de.html

MariaDB installieren und konfigurieren

```
$ brew install mariadb
$ mysql_secure_installation
```

MariaDB-Server starten

```
$ mysql.server start
```

bzw. wenn der MariaDB-Server automatisch im Rahmen jeden Neustarts gestartet werden soll:

```
$ brew services start mysql
```

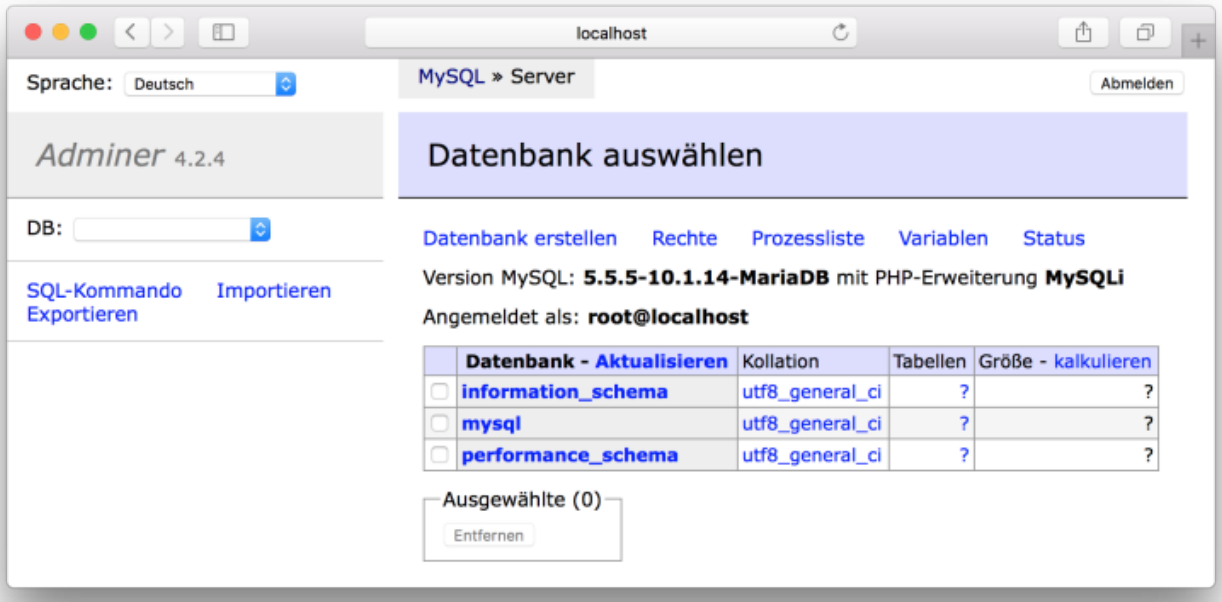
Adminer

Eine schlanke Alternative Zu phpmyadmin downloaden:

```
$ curl "https://www.adminer.org/static/download/4.2.4/adminer-4.2.4.php" >
~/Sites/adminer-4.2.4.php
```

2. Test

```
$ open -a Safari "http://localhost/~user/adminer-4.2.4.php"
```



<html></html> **Fertig?** <html></html>

Es kann durchaus vorkommen, dass man gerade dann feststellen darf, dass ein PHP-Modul fehlt, wenn man es braucht!

Seit 10.8 (Moutain Lion) soll pear (PHP Extension and Application Repository) nicht mehr Bestandteil von OS X sein?

pear installieren und konfigurieren

```
$ cd ~/Downloads
$ curl -O http://pear.php.net/go-pear.phar
$ php -d detect_unicode=0 go-pear.phar
$ sudo sh -c 'echo "include_path=".:/Users/user/pear/share/pear"" >>
/private/etc/php.ini'
$ sudo apachectl restart
```

In einem „jungfräulichen“ OS X hat haben rvm (Ruby Version Manager) und Homebrew lediglich „.bash_profile“ angepasst, weshalb es auch mein bevorzugter Ort für Anpassungen ist.

```
$ echo 'export PATH="$PATH:/Users/neupat75/pear/bin"' >> ~/.bash_profile
$ . ~/.bash_profile
```

pear testen bzw. aktualisieren

```
$ pear update-channels
$ pecl update-channels
$ pear upgrade
$ pear upgrade-all
```

Als zwischeinzeitlicher Ruby on Rails-„Bastler“ habe ich yaml lieben gelernt.

Ganz nebenbei ist es ein praktisches Beispiel zum Nachinstallieren eines PHP-Moduls.

Bevor überhaupt ein PHP-Modul gebaut werden kann, muss sichergestellt werden, dass die Xcode-Kommandozeilentools nebst Header-Dateien (.h) installiert sind und gefunden werden:

```
$ xcode-select --install
```

Ohne yaml-Bibliotheken kann auch kein yaml-PHP-Modul gebaut werden:

libyaml installieren

```
$ brew install libyaml
```

Da da /usr in einem OS X-System seit geraumer Zeit gegen Veränderungen besonders gut geschützt wird, müssen wir diesen Schutz kurzzeitig deaktivieren:

System Integrity Protection deaktivieren

Mac neustarten und sofort nach dem Apple-typischen Ton beim Booten eins Mac folgende Tastenkombination drücken, wonach das System von der Recovery-Partition starten sollte:

Cmd + R

Hier wählen wir aus dem Menü zuerst „Dienstprogramme“ und aus der Liste schließlich „Terminal“:

```
$ csrutil disable
$ reboot
```

Nach dem Neustart melden wir uns als „user“ an und öffnen wieder ein Terminal.

yaml-PHP-Modul bauen

```
$ sudo pecl install yaml
```

PHP-Konfiguration anpassen

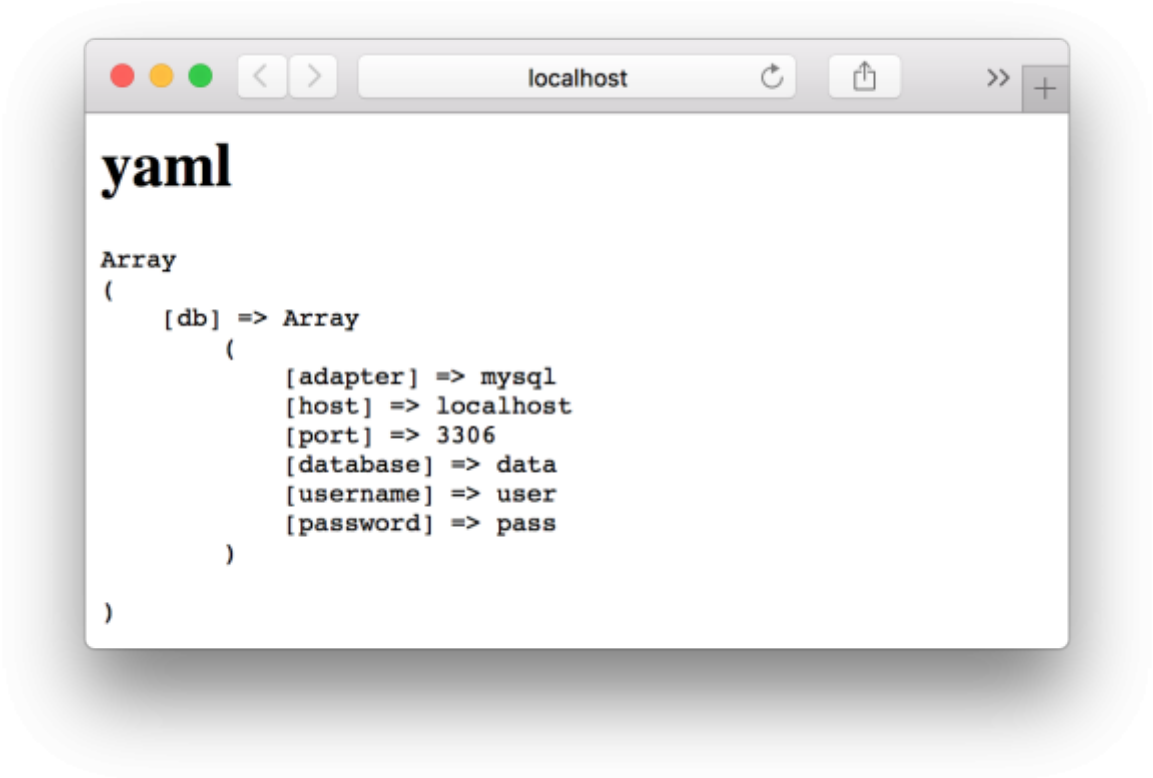
```
$ sudo sh -c 'echo "extension=yaml.so" >> /private/etc/php.ini'
$ sudo apachectl restart
```

3. Test

```
$ cat <<EOF > ~/Sites/db.cfg.yml
---
db:
  adapter: mysql
  host: localhost
  port: 3306
  database: data
  username: user
  password: pass
EOF
```

```
$ cat <<EOF > ~/Sites/db.inc.php
<?php
\$cfg = yaml_parse_file ( "./db.cfg.yml" );
echo "<html><head><title>yaml</title></head><body><h1>yaml</h1><pre>";
print_r( \$cfg );
echo "</pre></body></html>";
?>
EOF
```

```
$ open -a Safari "http://localhost/~user/db.inc.php"
```



System Integrity Protection reaktivieren

Reboot

Cmd + R

Dienstprogramme → Terminal

```
$ csrutil enable
$ reboot
```

<html></html> **Fertig!** <html></html>