

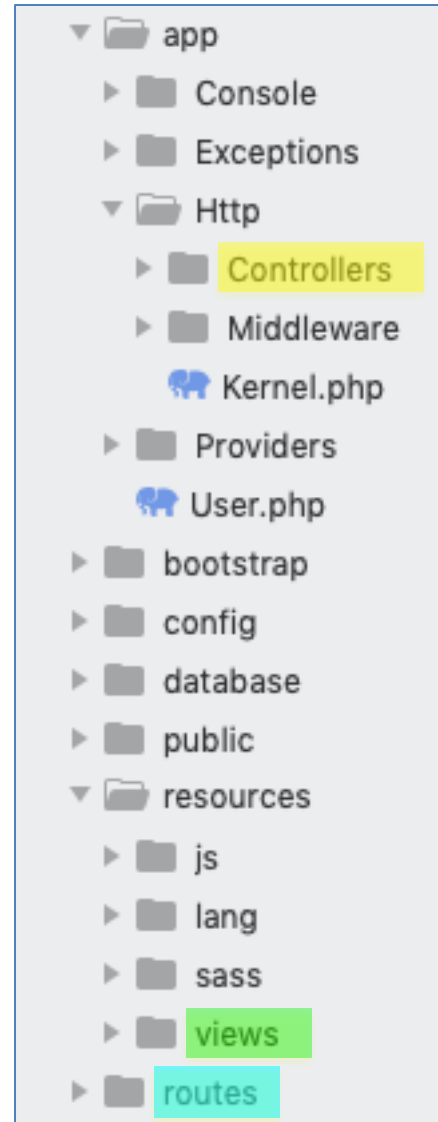
Componenti Laravel finora

Dei componenti base di Laravel, abbiamo finora introdotto: **controllers**, **views**, **routes**

Per completare, vanno introdotti i **modelli**, che consentono di accedere in modo uniforme, conciso ed elegante ai DB che fanno da *backend* dell'app Laravel

L'idea-base è che lo sviluppatore Laravel:

- non scriva (direttamente) query SQL,
- bensì codice PHP con oggetti appropriati, che corrispondano a tabelle e record del DB e i cui metodi abbiano effetti "automatici" sul DB



Il database backend

- Ma, prima ancora di iniziare a introdurre dei modelli, occorre attivare un database che faccia da "back end" della nostra applicazione Laravel
- La scelta tipica è un DB relazionale, per lo più *mysql*
- Nel seguito, si cerca di indicare concisamente come attivare e installare *mysql* e la relativa interfaccia grafica *phpmyadmin*
- Queste note sono tutt'altro che esaustive e puntano solo a fornire assistenza pratica, mantenendo l'enfasi su Laravel
- In alternativa, per spiegazioni comunque rapide, ma un po' più complete, si vedano [queste altre note](#)

Preliminari: preparare l'app Laravel per mysql

- Si parte, come al solito, con il wizard *laravel new*
- Tra i file generati, *.env* contiene profilo e configurazione dell'app Laravel
- In *.env* si noti il gruppo *DB*: il default è *mysql*, come qui a destra (ma si possono usare anche altri DB)

```
~ $ laravel new prog
```

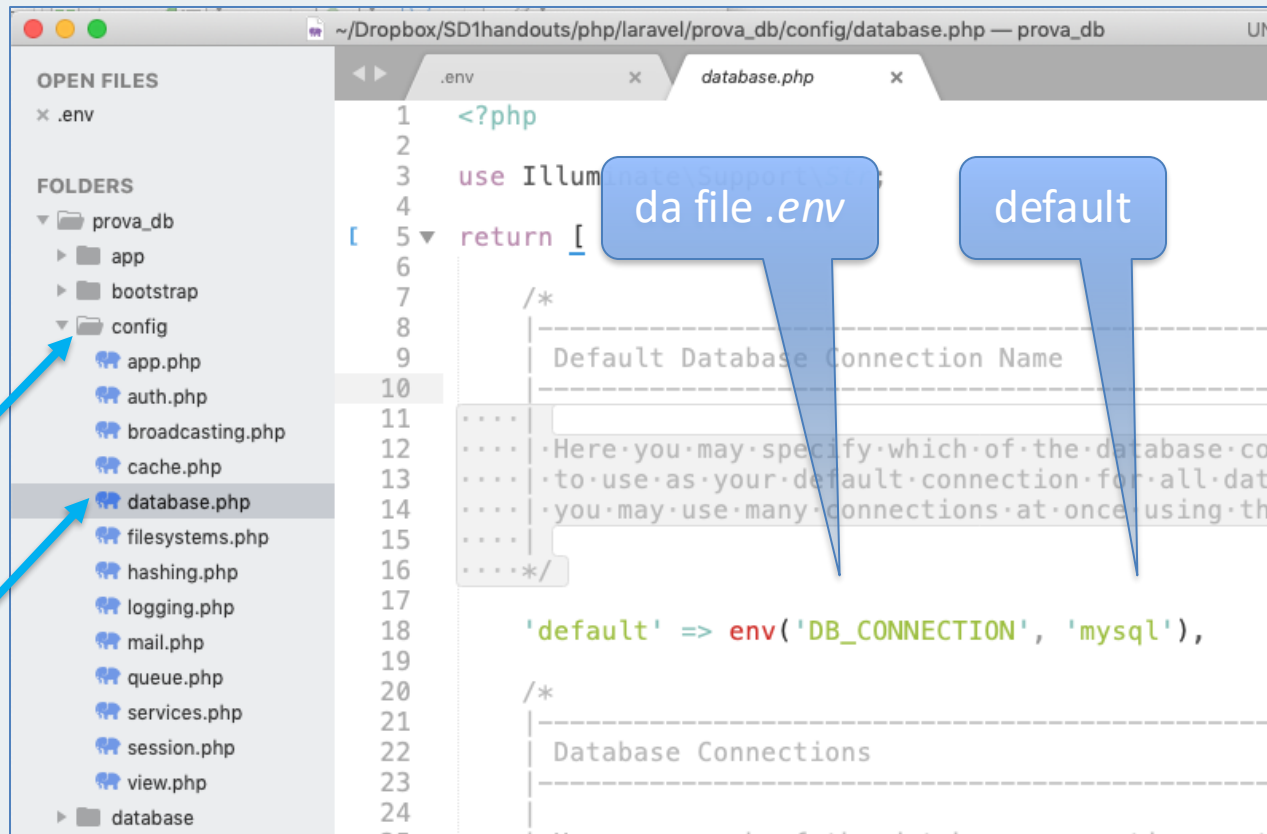
```
...
```

```
Application ready! Build something amazing.
```

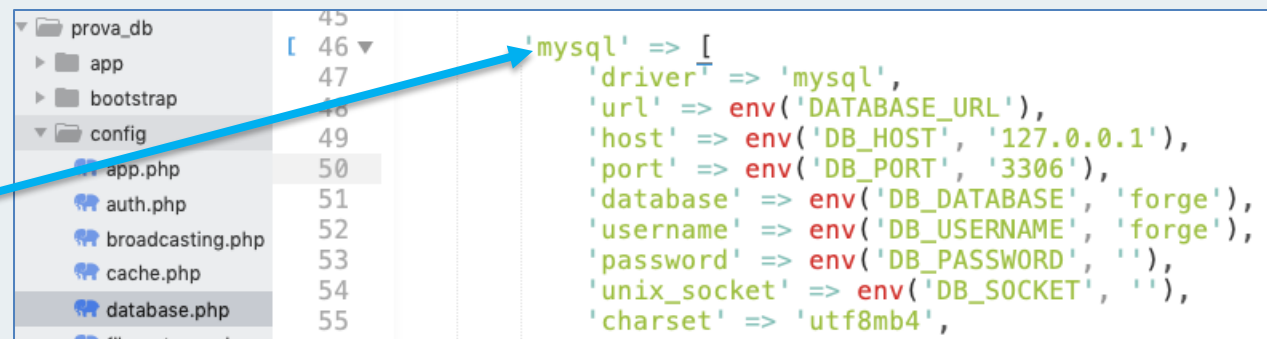
```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:Ujsh7YHoKF6bHBmeT
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=homestead
13 DB_USERNAME=homestead
14 DB_PASSWORD=secret
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
```

Laravel: lettura della configurazione per mysql

- Le variabili definite nel file `.env` vengono lette da codice `php` che, per lo più, sta nella dir `config`:
- Per il DB, in particolare, da `database.php`
- che sfrutta, per il nostro esempio, l'array di array `'mysql'`



```
1 <?php
2
3 use Illuminate\Support\Facades\Config;
4
5 return [
6
7     /*
8      * Default Database Connection Name
9      */
10
11     'default' => env('DB_CONNECTION', 'mysql'),
12
13     /*
14      * Database Connections
15      */
16
17     'mysql' => [
18         'driver' => 'mysql',
19         'url' => env('DATABASE_URL'),
20         'host' => env('DB_HOST', '127.0.0.1'),
21         'port' => env('DB_PORT', '3306'),
22         'database' => env('DB_DATABASE', 'forge'),
23         'username' => env('DB_USERNAME', 'forge'),
24         'password' => env('DB_PASSWORD', ''),
25         'unix_socket' => env('DB_SOCKET', ''),
26         'charset' => 'utf8mb4',
```



```
45
46 'mysql' => [
47     'driver' => 'mysql',
48     'url' => env('DATABASE_URL'),
49     'host' => env('DB_HOST', '127.0.0.1'),
50     'port' => env('DB_PORT', '3306'),
51     'database' => env('DB_DATABASE', 'forge'),
52     'username' => env('DB_USERNAME', 'forge'),
53     'password' => env('DB_PASSWORD', ''),
54     'unix_socket' => env('DB_SOCKET', ''),
55     'charset' => 'utf8mb4',
```

phpMyAdmin

- Per interagire con il DBMS relazionale *mysql* vi sono molti tool
- Useremo *phpMyAdmin*, che è open e... è anche una web app PHP!
- si può installare in molti modi, secondo la piattaforma
- se installate un pacchetto XAMPP..., avrete già *phpMyAdmin*
- oppure, si può scaricare come *.zip* dal sito o via GitHub o, essendo un app PHP, si può installare con *composer*, come mostrato qui:

```
~ $ composer create-project phpmyadmin/phpmyadmin
Installing phpmyadmin/phpmyadmin (4.8.5)
  - Installing phpmyadmin/phpmyadmin (4.8.5): Downloading (100%)
Created project in /Users/gp/phpmyadmin
...
~ $ cd /Users/gp/phpmyadmin
phpmyadmin $ composer update # aggiorna phpmyadmin
...
```

- ora si può eseguire *phpmyadmin* come app PHP **stand-alone**:

```
~ phpmyadmin $ php -S localhost:7777 # o altro port a piacere
```

Installare phpMyAdmin come pacchetto

- Un'altra opzione, per installare phpMyAdmin, è usare il gestore di pacchetti di sistema (brew-OSX / apt-debian...), magari anche per tutto lo stack *apache/php/phpmyadmin*

```
~ $ brew install httpd / apt install apache2    # httpd = apache  
~ $ brew/apt install php  
~ $ brew/apt install phpmyadmin  
# a questo punto occorre intervenire sui file di configurazione di apache/php, cf.  
# http://guide.debianizzati.org/index.php/Installare un ambiente LAMP: Linux, Apache2, SSL, MySQL, PHP5 - Stretch  
# https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-18-04  
# https://getgrav.org/blog/macos-mojave-apache-multiple-php-versions
```

- NB: le istruzioni alle URL sopra riguardano, a parte *phpmyadmin*, come installare php *come modulo* di Apache.
- In questo caso, la porta 80 è controllata da Apache che, se richiesto di servire una pagina php, si rivolge al modulo PHP

PHP come modulo di Apache

- In alternativa all'uso come interprete standalone, PHP può operare come modulo del Web server Apache
- Sull'argomento, le URL alla slide precedente forniscono ottime guide
- Una sintesi sulle configurazioni necessarie ce la propone il tool *brew* di OSX, come mostrato qui sotto, ma vale per Unix in generale:

```
$ brew info php
```

```
...
```

==> Caveats

To enable PHP in Apache add the following to httpd.conf and restart Apache:

```
LoadModule php7_module /usr/local/opt/php/lib/httpd/modules/libphp7.so
```

```
<FilesMatch \.php$>
```

```
    SetHandler application/x-httpd-php
```

```
</FilesMatch>
```

Finally, check `DirectoryIndex` includes `index.php`

```
DirectoryIndex index.php index.html
```

The `php.ini` and `php-fpm.ini` file can be found in: `/usr/local/etc/php/7.3/`

```
...
```

Phpmyadmin e PHP con Apache

- Se PHP gira come modulo di Apache, *phpmyadmin* va attivato intervenendo sui file di configurazione di Apache:

```
$ brew info phpmyadmin    # le stesse istruzioni si applicano a Apache/PHP su Linux
```

```
...
```

==> Caveats

To enable *phpMyAdmin* in Apache, add the following to `httpd.conf` and restart Apache:

```
Alias /phpmyadmin /usr/local/share/phpmyadmin
<Directory /usr/local/share/phpmyadmin/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    <IfModule mod_authz_core.c>
        Require all granted
    </IfModule>
    <IfModule !mod_authz_core.c>
        Order allow,deny
        Allow from all
    </IfModule>
</Directory>
```

Then open <http://localhost/phpmyadmin>

The configuration file is `/usr/local/etc/phpmyadmin.config.inc.php`

Avviare mysql come servizio

- *Mysql* o il suo fork *mariadb* girano come servizi
- In ambiente Windows conviene avviarlo all'interno di XAMPP/...
- In ambiente Unix si avviare il daemon da shell, così:

```
$ brew services run mariadb # OSX
```

```
$ sudo systemctl start mariadb # Linux
```

- Si presume ovviamente lo si fosse installato in precedenza, così:

```
$ brew install mariadb # OSX
```

```
$ apt install mariadb-server mariadb-client # Linux
```

In realtà installare su Ubuntu potrebbe avere dei prerequisiti, vedi:

<https://computingforgeeks.com/install-mariadb-10-on-ubuntu-18-04-and-centos-7/>

- L'uso di password per *root* (NB: *root* di *mysql*) è raccomandato in produzione, ma fonte di problemi in sviluppo, specie all'inizio
- Per eliminare la password o, in generale, ri-inizializzare i metadati di *mysql*: <https://dev.mysql.com/doc/refman/en/data-directory-initialization-mysqld.html>

Preparare il Database

- Prima di poter avviare l'applicazione dobbiamo creare e collegare un database, sia *esempio_db*, al nostro progetto.
- Creiamo quindi un nuovo database *esempio_db* e un nuovo utente con privilegi per *esempio_db*
- Non serve creare alcuna tabella con i dati, per ora – Laravel si occuperà di creare tabelle e relazioni di *esempio_db* automaticamente

```
$ mysql -u root # sotto Unix, potrebbe essere necessario premettere sudo (come spiegato prima)
Welcome to the MariaDB monitor.  Commands end with ; or \g. Your MariaDB connection id is 195...

MariaDB [(none)]> rehash -- per autocompletion! Meglio ancora in .my.cnf

MariaDB [(none)]> CREATE DATABASE esempio_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE USER 'pippo'@'localhost' IDENTIFIED BY 'pippo'; -- user e password
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> GRANT ALL ON esempio_db.* TO 'pippo'@'localhost'; -- occhio (anche sopra) agli apici
Query OK, 0 rows affected (0.005 sec)
```

mysql e Laravel

- Occorre creare "a mano" (o con *laravel new*), con *phpmyadmin* o *mysql*, il database di supporto per l'app; p.es. *prova_db* per l'app *prova*

```
$ mysql -u root # sotto Unix, potrebbe essere necessario premettere sudo (come spiegato prima)
MariaDB [(none)]> CREATE DATABASE prova_db;
Query OK, 1 row affected (0.001 sec)
```

- Per creare le tabelle si userà l'ambiente Laravel (non discusso qui)
- Il cuore dell'interazione tra *mysql* e *Laravel* è il file *.env* dell'app Laravel (generata automaticamente nella directory *prova*)

```
~/prova $ grep DB_ .env
grep DB_ .env
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

- Va modificato il nome del DB in *prova_db*
- user *root* senza password funziona se *mysql* non restringe questo accesso al super-user (come però accade ormai per default, v. slide precedenti)
- potrebbe essere meglio, se lo si è definito, (cf. slide precedenti) sfruttare un utente, p.es. *admin*, con i privilegi di *root*

```
DB_DATABASE=prova_db
DB_USERNAME=admin
DB_PASSWORD=admin
```

mysql e phpmyadmin

- Per accesso a *mysql* senza password, nella home di *phpmyadmin* si crei (o modifichi) il file *config.inc.php* (ultimo rigo qui a destra)
 - se, anziché stand-alone, PHP gira come modulo di Apache, il file è: *.../etc/phpmyadmin.config.inc.php*
- A questo punto, se si è già avviato il servizio *mysql*, si può interagire con esso con *phpmyadmin*
- NB: per problemi di autenticazione con *mysql*, cf. <https://stackoverflow.com/questions/11634084>

```
config.inc.php x
/* First server
*/
$i++;
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'cookie';
/* Server parameters */
$cfg['Servers'][$i]['host'] = 'localhost';
$cfg['Servers'][$i]['compress'] = false;
$cfg['Servers'][$i]['AllowNoPassword'] = true;
```



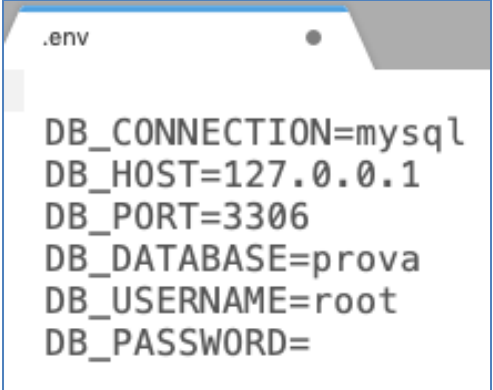
DB mysql di prova

Riprendiamo ora l'esempio
Laravel che ci aiuterà a
introdurre i modelli e una
vera app MVC

- creare con *phpmyadmin* un database
chiamato *prova*

I passi essenziali sono:

- creare con *laravel new* un nuovo progetto
prova_db, che ha per back-end il DB *prova*
- all'uopo modificare la sezione *DB_...* del file
.env del progetto, come mostrato qui a destra
- per maggiori dettagli, vedi [note su uso di mysql](#)



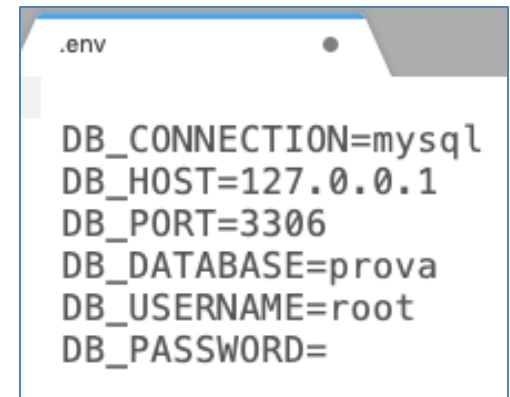
```
.env
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=prova
DB_USERNAME=root
DB_PASSWORD=
```

DB mysql di prova



I passi essenziali sono:

- creare con *phpmyadmin* un database chiamato *prova*
- creare con *laravel new* un nuovo progetto *prova_db*, che ha per back-end il DB *prova*
- all'uopo modificare la sezione *DB_...* del file *.env* del progetto, come mostrato qui a destra
- per maggiori dettagli, vedi [note su uso di mysql](#)



DB mysql di prova

Riprenderemo ora l'esempio Laravel che ci aiuterà a introdurre i modelli e una vera app MVC.



```
.env
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=prova
DB_USERNAME=root
DB_PASSWORD=
```