**Documentation**

# **Laravel**

## **clone laravel - github repository**

1. git clone

2. composer install (in cmd)

3. npm install (in cmd)

4. .env.example file (make a copy and delete the .example -> .env)

5. php artisan key:generate (in cmd)

6. create empty database in Mysql

7. in .env file fill DB\_HOST, DB\_DATABASE, DB\_USERNAME, DB\_PASSWORD

8. php artisan migrate –force (in cmd - creates tables in database)

## **Views**

### **layout and content of pages**

**path:** resources/views

**folder “includes”** – contains navbar, footer and head (can be put into a page with @include(‘includes.navbar’)

**folder layout** – contains the basic layout of the pages, with the includes and the content

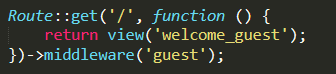
@yield('content') – here the content of different pages is displayed

Folders with names of specific tables or roles include crud pages (create, edit, delete, show) for the named table(s)

## **Routing**

**path:** routes/web.php

Route for given view (welcome\_guest) with link abbreviation (‘/’) and the authorization information (middleware)



Route for a Controller where routes and functions for CRUD pages are



## **Controllers**

**path:** app/Http/Controllers

**folder auth** – Controller for register, login, logout, password and verification is situated

Name of Controllers displays the tables that the content (CRUD) uses

At the top there are the links of the Models and Files the Controller refers to (uses)

The class is for displaying the various CRUD pages as well as the CRUD functions itself.

#### CREATE

<tablename>::create([

‘attribute’ => [‘name of input field’],

‘attribute’ => [‘name of input field’],

]);

#### UPDATE

<tablename>::where('attribute', $param)->update($request->except('\_token', '\_method'));

#### DESTROY (=delete)

if(<tablename>::destroy($id)) {

return redirect(‘<view>’);

} else {

return redirect(‘<view>’);

}

<tablename>::all() – select all of given table

<tablename>::find($param) – find specific row (according to parameter) of given table

## **FILE .env**

Here the database connection is made

DB\_CONNECTION=mysql

DB\_HOST=localhost

DB\_DATABASE=laratest

DB\_USERNAME=root

DB\_PASSWORD=

## **SASS**

path: resources/sass

with command **npm run** in cmd sass is auto compiled

## **Migrations**

Migrations are files with information of the database tables

**path:** database/migrations

## **Middleware**

Here the authorizations are set

**path:** app/Http/middleware

kernel -> register middleware

## **JOINS AND PIVOT TABLE**

Accessing Data in a table that is connected with a FK (Join)

Accessing data in a table that is connected through a reference table, many to many connection (Pivot)

**In Model**

* For join

public function <function name> (){

return $this->belongsTo('App\<Model of foreign table>', '<FK>’);

}

* For pivot

public function <function name> () {

return $this->belongsToMany('App\\<Model of foreign table>’, '<name of table as in mysql>', '<FK to first table>', '<FK to second table>')->withPivot('<name of field that is to be accessed>');

}

There is a need for a separate Controller, name convention used first table + To + second Table (CamelCase)

Used tables in project:

* ClientsToConsulting
* CoursesToSkill
* ProfilesToSkill

**Routing**

Route::post('< > '<controller name>@<function name>')

->name('<name of route>');

Route::get('< > '<controller name>@<function name>')

->name('<name of route>');

Route::delete('< > '<controller name>@<function name>')

->name('<name of route>');

**Access in Views**

* For single data

<variable set for starting table> 🡪 <call function()> 🡪 <name of field>

* For multiple data

@foreach(<variable set for starting table> 🡪 <call function()> 🡪 get() as <new variable name>)

# **CMD**

**php artisan list** – list all commands

**php artisan serve** – start server

**php artisan make:controller <controller name> -r -m <Model name>**

creates controller as well as resources and a model

**php artisan make:model**

create a new Eloquent model class

**php artisan make:middleware**

creates a middleware

**php artisan migrate:fresh**

drop all tables and re-run all migrations

**php artisan migrate:generate**

generate a migration from an existing table structure.

**php artisan migrate:refresh**

reset and re-run all migrations

**help to a special command**

php artisan -h <command>