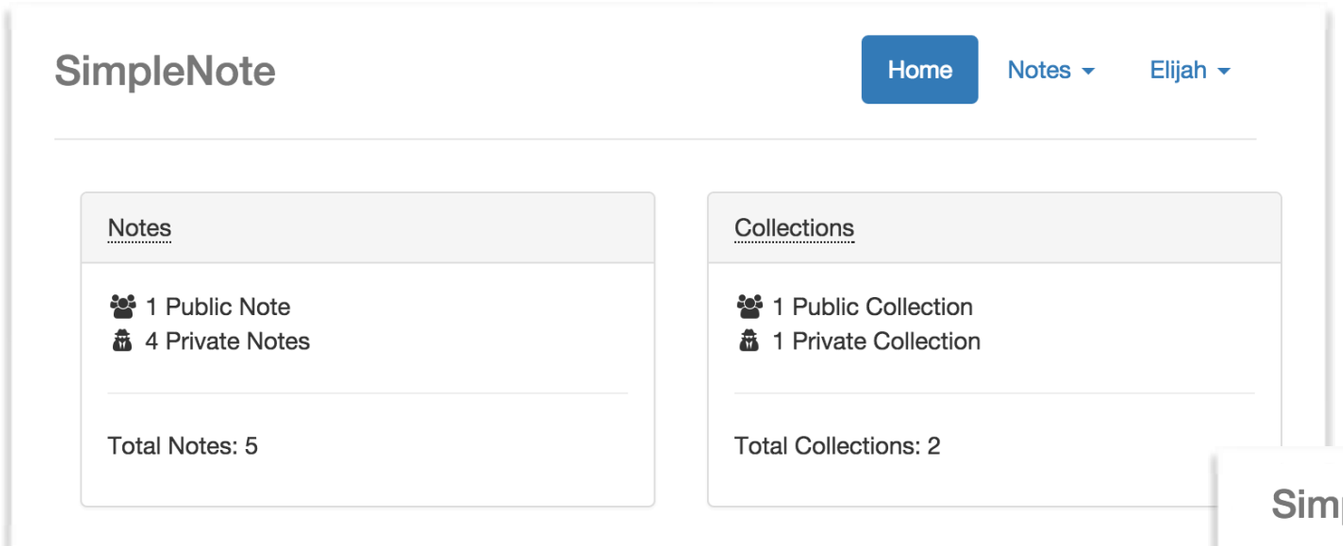
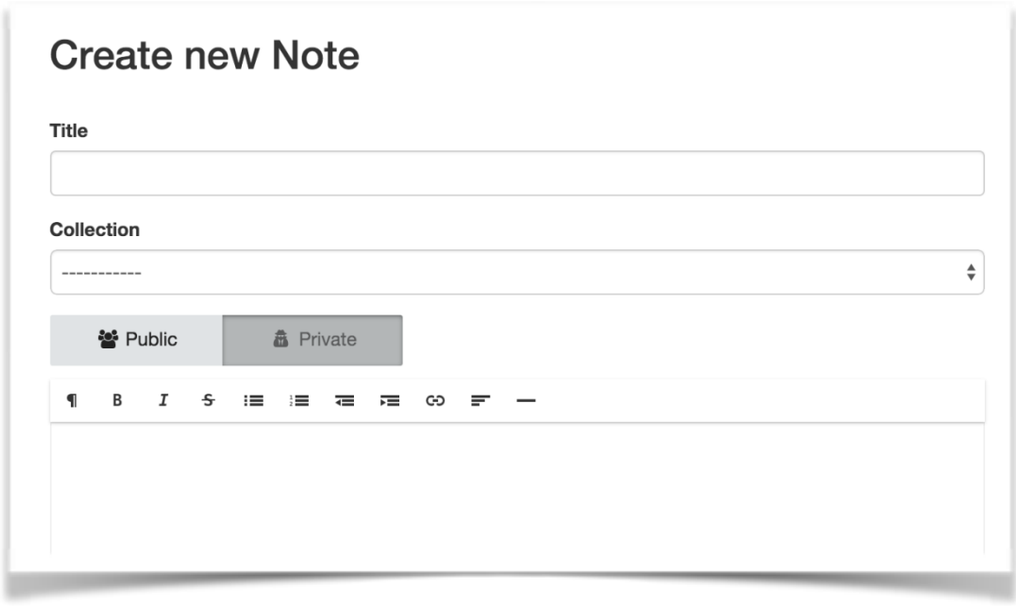
Laravel 5.0

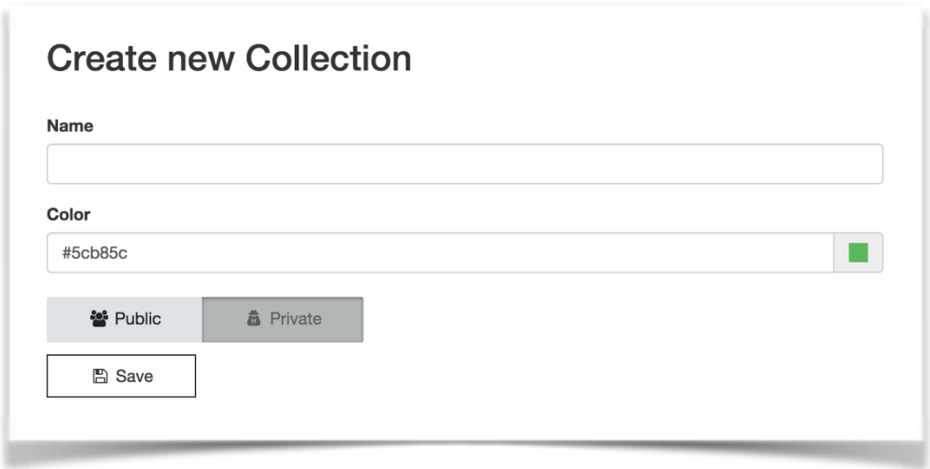
PHP Web Application

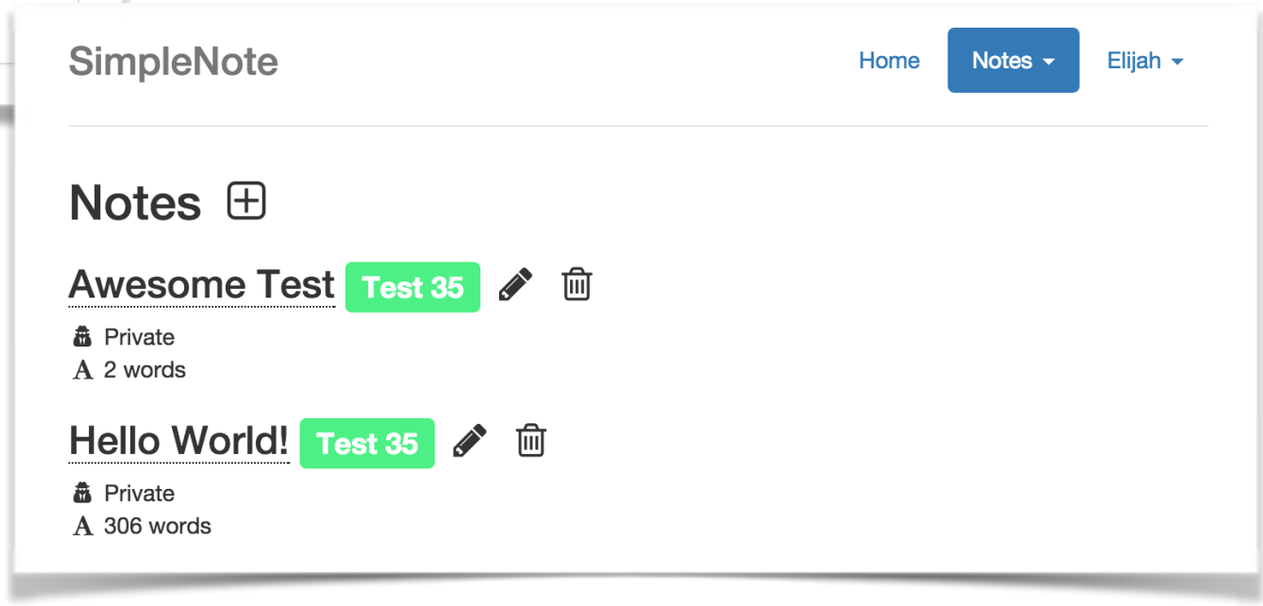
End Product

- Simple Note Taking Application
- Organize Notes in Collections









Implementation

- Laravel 5.0 (PHP)
- MySQL (Database)
- Cloud9 (Web hosting)

Laravel 5.0

- http://laravel.com/
- PHP Web Application Framework

MySQL

- http://www.mysql.com
- SQL Database

Cloud9

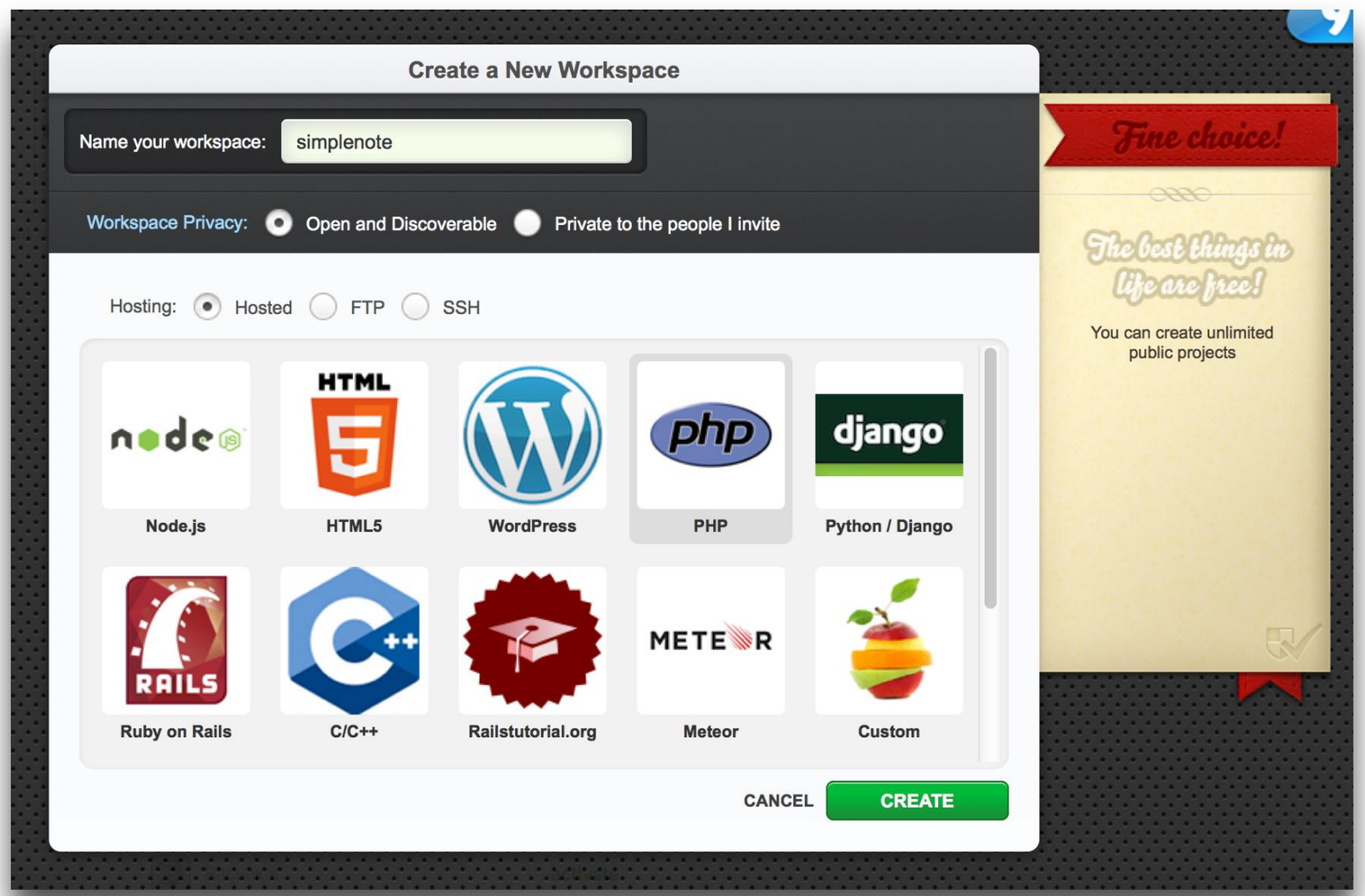
- https://c9.io
- Online IDE and web hosting for small/public projects

Getting Started

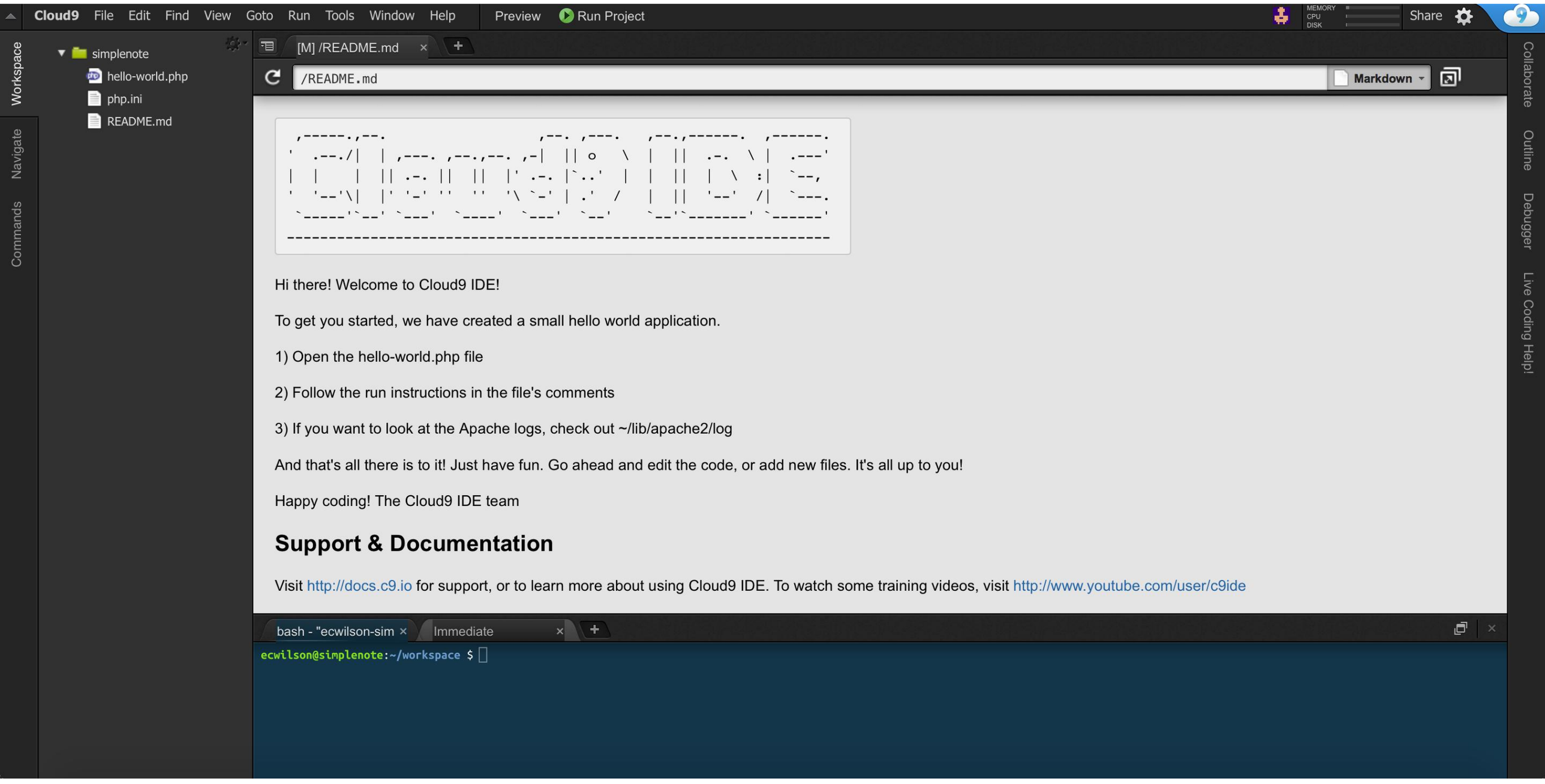
- Create free Cloud9 account

Create Cloud9 Workspace

- Name it "simplecontacts"
- Select "PHP"



Open Workspace



Create Laravel Project

```
rm *
sudo composer self-update
composer create-project laravel/laravel ./laravel --prefer-dist
shopt -s dotglob
mv laravel/* ./
rm -rf laravel
sudo composer update
```

Modify Apache Config

```
sudo pico /etc/apache2/sites-enabled/001-cloud9.conf

// Change this line

DocumentRoot /home/ubuntu/workspace

// To following

DocumentRoot /home/ubuntu/workspace/public
```

Install Dependencies

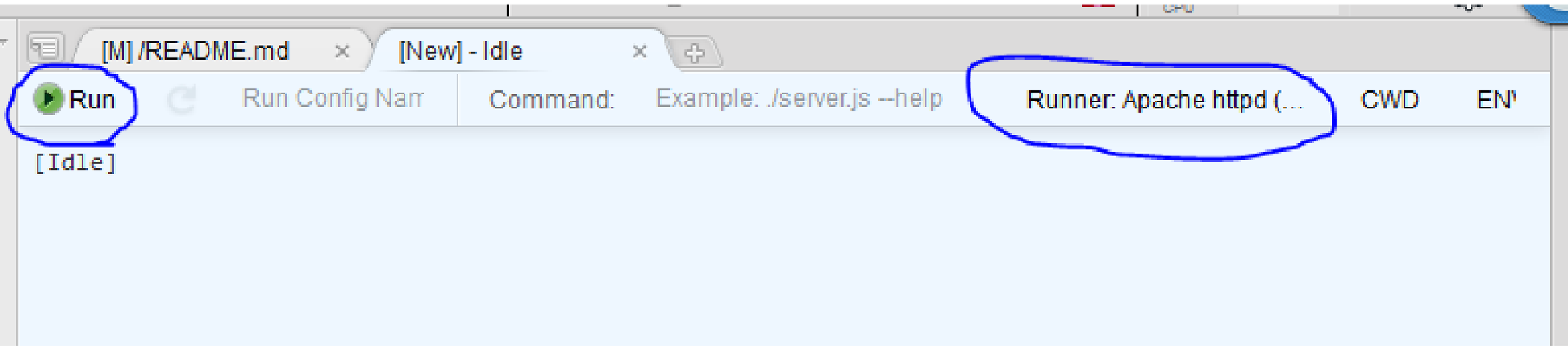
```
sudo composer update
```

Compile Error?

```
mv ./vendor/compiled.php ./vendor/compiledold.php

php artisan down

php artisan clear-compiled

php artisan optimize

php artisan up

sudo composer update
```

Apache Runner

- Create a new tab & select "New Run Configuration"
- On the right side, select "Apache httpd (PHP, HTML)"
- Press "Run"



Database Setup

```
$ pico .env

DB_HOST=localhost

DB_DATABASE=simplecontacts

DB_USERNAME=doge

DB_PASSWORD=wow
```

Database Setup (cont.)

- sudo mysql
- CREATE DATABASE simplecontacts;
- CREATE USER 'doge' IDENTIFIED BY 'wow';
- GRANT ALL ON simplecontacts.* TO 'doge';
- exit

Create Contact db Table

```
$ php artisan make:migration CreateContactTable
```

This will create the file:

database/migrations/2015_xx_xx_xxxxxx_CreateContactTable.php

Insert the code on the following slide.

```php
<?php

use Illuminate₩Database₩Schema₩Blueprint;
use Illuminate₩Database₩Migrations₩Migration;

class CreateContactTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('contact', function($table)
        {
            $table->increments('id');
            $table->string('fname');
            $table->string('lname');
            $table->string('email')->nullable();
            $table->string('phone')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('contact');
    }
}
```

Migrate Database

- php artisan migrate
- For more info on migrations see http://laravel.com/docs/5.1/migrations#introduction

Seed Database with Data

```
$ php artisan make:seeder ContactTableSeeder
```

This will create the file:

database/seeds/ContactTableSeeder.php

Insert the code on the following slide.

```php
<?php

use Illuminate₩Database₩Seeder;

class ContactTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('contact')->insert([
            'fname' => 'Walter',
            'lname' => 'White',
            'email' => 'no-reply@savewalterwhite.com',
            'phone' => '(555) 555-5555'
        ]);
    }
}
```

- Insert the following code into app/database/seeds/DatabaseSeeder.php

```php
<?php

use Illuminate¥Database¥Seeder;
use Illuminate¥Database¥Eloquent¥Model;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Model::unguard();

        $this->call(ContactTableSeeder::class);

        Model::reguard();
    }
}
```

Seed Database with Data cont.

- `$ php artisan db:seed`

ContactController

- Controllers receive requests directed to them via routes.php
- Returns a view or redirect to another route

```
$ php artisan make:controller ContactController
```

routes.php

- We will route the application root to the ContactsController which will handle all additional routing
- Replace the code in app/Http/routes.php with the following:

```php
<?php

/*
|--------------------------------------------------------------------------
| Application Routes
|--------------------------------------------------------------------------
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::controller('/', 'ContactController');
```

Contact Model

- $ php artisan make:model Contact
- Insert the following code into the Contact class in app/Contact.php

```
public $timestamps = false;

/**
 * The database table used by the model.
 *
 * @var string
 */
protected $table = 'contact';

/**
 * The attributes that are mass assignable.
 *
 * @var array
 */
protected $fillable = ['fname', 'lname', 'email', 'phone'];
```

Contacts View

- This will display a tabular listing of all contacts in the database
- Create the file resources/views/contacts.blade.php
- Insert the code on the following slides

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Contacts</title>
        <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
        <style>
            .container
            {
                padding-top: 1em;
            }
            table td
            {
                border-top: none !important;
            }
        </style>
    </head>
    <body>
         <div class="container">
            <div class="row">
                <p>
                    <a href="create" class="btn btn-success">New Contact</a>
                </p>
            </div>
            <div class="row">
            <table class="table">
                <thead>
                    <tr>
                        <th>Name</th>
                        <th>Email</th>
                        <th>Phone</th>
                    </tr>
                </thead>
                <tbody>
                    <?php foreach ($contacts as $contact): ?>
                    <tr>
                        <td>{{$contact->lname. ', '. $contact->fname}}</td>
                        <td>{{$contact->email}}</td>
                        <td>{{$contact->phone}}</td>
                        <td>
                            <a href="edit/{{$contact->id}}" class="btn btn-success">Edit</a>
                            <a href="delete/{{$contact->id}}" class="btn btn-danger">Delete</a>
                        </td>
                    </tr>
                    <?php endforeach; ?>
                </tbody>
            </table>
            </div>
        </div>
    </body>
</html>
```

Add Contacts Listing to ContactController

- Remove all methods from the ContactController class in app/Http/Controllers/ContactController.php

- Add the following method to the class:

```
/**
 * Display a listing of contacts.
 *
 * @return Response
 */
public function getIndex()
{
    return view('contacts', [
        'contacts' => ₩App₩Contact::orderBy('lname')->get()
    ]);
}
```

- You should now be able to see the single contact we previously added to the database when you run the application

- You will get errors if you attempt to create, edit, or delete. Don’t worry, we’ll fix that shortly!

Create the Create View

- Create the file resources/views/create.blade.php
- Insert the code on the following slide

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>New Contact</title>
        <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
    </head>
    <body>
        <div class="container">
            <h1>Create New Contact</h1>
            <form method="post" class="form-horizontal">
                <input type="hidden" name="_token" value="{{ csrf_token() }}">
                <div class="form-group {{!empty($fnameError) ? 'has-error' : ''}}">
                    <label class="col-xs-2 control-label">First Name</label>
                    <div class="col-xs-6">
                        <input name="fname" type="text" class="form-control" value="{{$fname or ''}}" placeholder="First Name">
                    </div>
                    <p class="col-xs-4 help-block">{{$fnameError or ''}}</p>
                </div>
                <div class="form-group {{!empty($lnameError) ? 'has-error' : ''}}">
                    <label class="col-xs-2 control-label">Last Name</label>
                    <div class="col-xs-6">
                        <input name="lname" type="text" class="form-control" value="{{$lname or ''}}" placeholder="Last Name">
                    </div>
                    <p class="col-xs-4 help-block">{{$lnameError or ''}}</p>
                </div>
                <div class="form-group {{!empty($emailError) ? 'has-error' : ''}}">
                    <label class="col-xs-2 control-label">Email</label>
                    <div class="col-xs-6">
                        <input name="email" type="text" class="form-control" value="{{$email or ''}}" placeholder="Email">
                    </div>
                    <p class="col-xs-4 help-block">{{$emailError or ''}}</p>
                </div>
                <div class="form-group {{!empty($phoneError) ? 'has-error' : ''}}">
                    <label class="col-xs-2 control-label">Phone</label>
                    <div class="col-xs-6">
                        <input name="phone" type="text" class="form-control" value="{{$phone or ''}}" placeholder="Phone Number">
                    </div>
                    <p class="col-xs-4 help-block">{{$phoneError or ''}}</p>
                </div>
                <input type="submit" value="Create Contact">
            </form>
        </div>
    </body>
</html>
```

Add Create Page to ContactController

- Add the following methods to the ContactController class in app/Http/Controllers/ContactController.php

```
/**
 * Show the form for creating a new contact.
 */
public function getCreate()
{
    return view('create');
}

/**
 * Save the contact to the database.
 */
public function postCreate(Request $request)
{
    ₩App₩Contact::create($request->all())->save();
    return redirect('/');
}
```

Add Delete Functionality to ContactController

- Add the following method to the ContactController class in app/Http/Controllers/ContactController.php

```
/**
 * Delete the contact with the specified ID.
 */
public function getDelete($id)
{
    ₩App₩Contact::destroy($id);
    return redirect('/');
}
```

Create the Edit View

- Create the file resources/views/edit.blade.php
- Insert the code on the following slide

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Edit Contact</title>
        <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
    </head>
    <body>
        <div class="container">
                <h1>Edit Contact</h1>
                <form method="post" class="form-horizontal">
                    <input type="hidden" name="_token" value="{{ csrf_token() }}">
                    <input type="hidden" name="id" value="{{$id}}">
                    <div class="form-group {{!empty($fnameError) ? 'has-error' : ''}}">
                        <label class="col-xs-2 control-label">First Name</label>
                        <div class="col-xs-6">
                            <input name="fname" type="text" class="form-control" value="{{$fname or ''}}" placeholder="First Name">
                        </div>
                        <p class="col-xs-4 help-block">{{$fnameError or ''}}</p>
                    </div>
                    <div class="form-group {{!empty($lnameError) ? 'has-error' : ''}}">
                        <label class="col-xs-2 control-label">Last Name</label>
                        <div class="col-xs-6">
                            <input name="lname" type="text" class="form-control" value="{{$lname or ''}}" placeholder="Last Name">
                        </div>
                        <p class="col-xs-4 help-block">{{$lnameError or ''}}</p>
                    </div>
                    <div class="form-group {{!empty($emailError) ? 'has-error' : ''}}">
                        <label class="col-xs-2 control-label">Email</label>
                        <div class="col-xs-6">
                            <input name="email" type="text" class="form-control" value="{{$email or ''}}" placeholder="Email">
                        </div>
                        <p class="col-xs-4 help-block">{{$emailError or ''}}</p>
                    </div>
                    <div class="form-group {{!empty($phoneError) ? 'has-error' : ''}}">
                        <label class="col-xs-2 control-label">Phone</label>
                        <div class="col-xs-6">
                            <input name="phone" type="text" class="form-control" value="{{$phone or ''}}" placeholder="Phone Number">
                        </div>
                        <p class="col-xs-4 help-block">{{$phoneError or ''}}</p>
                    </div>
                    <input type="submit" value="Update Contact">
                </form>
                <div class="colspan">

                </div>
        </div>
    </body>
</html>
```

Add Edit Page to ContactController

- Add the following methods to the ContactController class in app/Http/Controllers/ContactController.php

```
public function getEdit($id = 0)
{
    try {
        $contact = ₩App₩Contact::findOrFail($id);
    }
    catch(₩Exception $e) {
        return redirect('/');
    }
    return view('edit', $contact->attributesToArray());
}

public function postEdit(Request $request)
{
    $id = $request->input('id');
    $contact = ₩App₩Contact::find($id);
    $contact->fill($request->all());
    $contact->update();
    return redirect('/');
}
```

- Now, assuming you did everything correctly (no refunds!), you should have a fully functional CRUD application. Hooray!!!

- There are still some nice features that we can add to make it a little more robust, such as form validation and sorting contacts.

Adding form Validation

- $ php artisan make:request ContactFormRequest
- This will create the file app/Http/Requests/ContactFormRequest.php
- Insert the code on the following slide

```php
<?php

namespace App₩Http₩Requests;

use App₩Http₩Requests₩Request;

class ContactFormRequest extends Request
{

    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        extract($this->all());
        return [
            'fname' => 'required|unique:contact,fname,'. (isset($id) ? $id : 'NULL'). ',id,lname,'. $lname,
            'lname' => 'required|unique:contact,lname,'. (isset($id) ? $id : 'NULL'). ',id,fname,'. $fname,
            'email' => 'email',
            'phone' => 'phone'
        ];
    }

    public function messages()
    {
        return [
            'required' => 'Required field',
            'email' => 'That is not a valid email address',
            'phone' => 'A phone number must be of the form (###) ###-####',
            'unique' => 'Full name must be unique',
        ];
    }
}
```

- Insert the following line near the top of app/Providers/AppServiceProvider next to the existing use clause.

```
use Validator;
```

- Insert the following code into the boot() method

```
Validator::extend('email', function($attribute, $value, $parameters) {
    return filter_var($email,FILTER_VALIDATE_EMAIL);
});

Validator::extend('phone', function($attribute, $value, $parameters) {
    $phoneFilter = '/₩(₩d{3}₩) ₩d{3}-₩d{4}/';
    return filter_var($value,FILTER_VALIDATE_REGEXP,array('options'=>array('regexp'=>$phoneFilter)));
});
```

- In app/Http/Controllers/ContactController, modify the postCreate() and postEdit() function headers as below

```
public function postCreate(Requests₩ContactFormRequest $request)
public function postEdit(Requests₩ContactFormRequest $request)
```

- Add the following PHP code to the top of the resources/views/create.blade.php and resources/views/edit.blade.php files.

```
<?php
    $fnameError = $errors->first('fname');
    $lnameError = $errors->first('lname');
    $emailError = $errors->first('email');
    $phoneError = $errors->first('phone');

    $fname = isset($fname) ? $fname : old('fname');
    $lname = isset($lname) ? $lname : old('lname');
    $email = isset($email) ? $email : old('email');
    $phone = isset($phone) ? $phone : old('phone')
?>
```

Adding Sort Functionality

- Add the following code in an appropriate location in resources/views/contacts.blade.php

```
<?php $order = isset($order) ? $order : 'lname' ?>
<form id="order" method="post">
    <input type="hidden" name="_token" value="{{ csrf_token() }}">
    <label>Sort by:</label>
    <select name="order" onchange="submitOrder()">
        <option value="lname" {{$order == 'lname' ? 'selected' : ''}}>Name</option>
        <option value="email" {{$order == 'email' ? 'selected' : ''}}>Email</option>
        <option value="phone" {{$order == 'phone' ? 'selected' : ''}}>Phone</option>
    </select>
</form>
```

- Add the following script to the HTML header

```
<script type="text/javascript">
    function submitOrder() {
        document.getElementById("order").submit();
    }
</script>
```

Adding Sort Functionality cont.

- Add the following method to the ContactsController class

```
/**
 * Display a list of sorted contacts.
 */
public function postIndex(Request $request)
{
    $order = $request->input('order');
    return view('contacts', [
        'order' => $order,
        'contacts' => ₩App₩Contact::orderBy($order)->get()
    ]);
}
```